Postel

Network Working Group
Request for Comments: 1095

U. Warrier
Unisys Corporation
L. Besaw
Hewlett-Packard
April 1989

The Common Management Information Services and Protocol over TCP/IP (CMOT)

Table of Contents

1. Status of this Memo 3
2. Introduction 4
Part I: Concepts and Models 7
3. The OSI Management Framework 7
3.1. Architectural Overview 7
3.2. Management Models 8
3.2.1. The Organizational Model 8
3.2.2. The Functional Model 8
3.2.3. The Information Model 9
3.3. ISO Application Protocols 9
3.3.1. ACSE 10
3.3.2. ROSE 10
3.3.3. CMISE 10
3.3.3.1. Management Association Services 11
3.3.3.2. Management Notification Services 12
3.3.3.3. Management Operation Services 12
4. The CMOT Architecture 13
4.1. Management Models 13
4.1.1. The Organizational Model 13
4.1.2. The Functional Model 14
4.1.3. The Information Model 14
4.2. Protocol Architecture 14
4.2.1 The Lightweight Presentation Layer 15
4.2.2 The Quality of Transport Service 16
4.3. Proxy Management 17
4.4. Directory Service 18
5. Management Information 18
5.1. The Structure of Management Information 19
5.1.1. The ISO SMI 19
5.1.1.1. Managed Objects and Attributes 19
5.1.1.2. Management Information Hierarchies 20
5.1.1.2.1 The Registration Hierarchy 20
5.1.1.2.2. The Containment Hierarchy 20
5.1.1.2.3. The Inheritance Hierarchy 22
5.1.2. The Internet SMI 22
5.2. The Management Information Base 23

5.3. An Interpretation of the Internet SMI 24
5.3.1. Object Class and Attributes 25
5.3.1.1. Object Class 25
5.3.1.2. Attribute Identifier 26
5.3.2. Management Information Hierarchies 26
5.3.2.1. The Registration Hierarchy 26
5.3.2.2. The Containment Hierarchy 26
5.3.2.3. The Inheritance Hierarchy 28
5.4. Scoping, Filtering, and Synchronization 28
5.4.1. Scoping 28
5.4.2. Filtering 29
5.4.3. Synchronization 29
5.4.4. Linked Replies 29
5.5. Accessing Tables 29
5.5.1. Accessing Whole Tables 30
5.5.2. Accessing Table Entries 30
Part II: Protocol Agreements 32
6. CMOT Protocol Overview 32
6.1. The CMOT Protocol Suite 32
6.2. Conformance Requirements 33
6.3. Abstract Syntax Notation 33
7. Common Management Information Service Element 34
7.1. CMIS Services 34
7.1.1. CMIS Services Overview 34
7.1.2. Functional Units 34
7.1.3. Functional Unit Groups 36
7.1.4. M-INITIALISE Parameters 37
7.1.4.1. Functional Units 37
7.1.4.2. User Information 39
7.1.4.3. Access Control 39
7.2. Supporting Services 39
7.3. CMIP Agreements 39
7.3.1. Invoke Identifier 39
7.3.2. Object Class 40
7.3.3. Object Instance 40
7.3.4. Access Control 41
7.3.5. Synchronization 41
7.3.6. Scope 41
7.3.7. Filter 41
7.3.8. Attribute Identifier 42
7.3.9. Event Type Identifier 42
7.3.10. Action Type Identifier 42
7.3.11. Time Fields 43
7.3.12. Response PDUs 43
7.3.13. Error PDUs 43
8. Association Control Service Element 43
8.1. ACSE Services 44
8.2. Supporting Services 44

8.3. ACSE Protocol 45
8.3.1. Application Context Name 45
8.3.2. User Information 45
8.3.3. Presentation Service Parameters 46
9. Remote Operations Service Element 46
9.1. ROSE Services 46
9.2. Supporting Services 47
9.3. ROSE Protocol 47
9.3.1. Operation Class 47
9.3.2. Priority 48
10. Lightweight Presentation 48
10.1. Lightweight Presentation Services 48
10.2. Supporting Services 48
10.3. Lightweight Presentation Protocol 49
11. Acknowledgements 49
12. References 49
Appendix A - The CMOT Group 52
Appendix B - Management Information Summary 53
Appendix C - Sample Protocol Exchanges 60

1. Status of this Memo

This memo defines a network management architecture that uses the International Organization for Standardization's (ISO) Common Management Information Services/Common Management Information Protocol (CMIS/CMIP) in a TCP/IP environment. This architecture provides a means by which control and monitoring information can be exchanged between a manager and a remote network element. In particular, this memo defines the means for implementing the Draft International Standard (DIS) version of CMIS/CMIP on top of Internet transport protocols for the purpose of carrying management information defined in the Internet-standard management information base. DIS CMIS/CMIP is suitable for deployment in TCP/IP networks while CMIS/CMIP moves toward becoming an International Standard. Together with the relevant ISO standards and the companion RFCs that describe the initial structure of management information and management information base, these documents provide the basis for a comprehensive architecture and system for managing TCP/IP-based internets, and in particular the Internet.

The Internet Activities Board (IAB) has designated two different network management protocols with the same status of "Draft Standard" and "Recommended".

The two protocols are the Common Management Information Services and Protocol over TCP/IP (CMOT) (this memo) and the Simple Network Management Protocol (SNMP) [4].

The IAB intends each of these two protocols to receive the attention of implementers and experimenters. The IAB seeks reports of experience with these two protocols from system builders and users.

By this action, the IAB recommends that all IP and TCP implementations be network manageable (e.g., implement the Internet MIB [3], and that implementations that are network manageable are expected to adopt and implement at least one of these two Internet Draft Standards.

Distribution of this memo is unlimited.

2. Introduction

As reported in RFC 1052, "IAB Recommendations for the Development of Internet Network Management Standards" [1], the Internet Activities Board (IAB) has directed the Internet Engineering Task Force (IETF) to coordinate the work of three working groups in the area of network management. First, the MIB working group was charged with the specification and definition of elements to be included in the Management Information Base (MIB). Second, the SNMP working group was charged with defining the modifications to the Simple Network Management Protocol (SNMP) necessary to accommodate the short-term needs of the network vendor and operations communities. Third, the Netman working group was directed to meet the longer-term needs of the Internet community by developing a network management system based on ISO CMIS/CMIP. Both the Netman working group and the SNMP working group were directed to align their work with the output of the MIB working group in order to ensure compatibility of management information between the short-term and long-term approaches to the management of TCP/IP-based internets. This will enable a smooth transition from the short-term protocol (SNMP) to the long-term protocol (CMIP).

The MIB working group has produced two memos. RFC 1065 [2] defines the Structure of Management Information (SMI) that is necessary for naming and defining managed objects in the MIB. RFC 1066 [3] defines the list of managed objects contained in the initial TCP/IP MIB. The SNMP working group has produced a memo [4] giving the protocol specification for SNMP and providing the SNMP protocol-specific interpretation of the Internet-standard MIB defined in RFC 1066.

This memo is the output of the Netman working group. As directed by the IAB in RFC 1052, it addresses the need for a long-term network management system based on ISO CMIS/CMIP. The network management approach of using ISO protocols in a TCP/IP environment to manage TCP/IP networks can be described as "CMIP Over TCP/IP" (CMOT). This memo specifies the CMOT architecture and the protocol agreements

necessary to implement CMIP and accompanying ISO protocols over the TCP and UDP transport protocols. In addition, this memo provides an interpretation of RFC 1066 that makes it possible to use CMIP to convey management information defined in the Internet-standard MIB.

There is widespread vendor support for the CMOT approach to network management. This is amply shown by the Netman demonstration of prototype CMOT implementations at the Interop '88 TCP/IP Interoperability Conference. The demonstration also showed the feasibility and power of the CMIS/CMIP framework for multivendor network management. Now that CMIS/CMIP has been voted a Draft International Standard (DIS), many vendors feel that the ISO standard has become a stable basis for product development. The clear need to standardize this development has led to the present profile of CMIP. It is expected that this profile will not change while the ISO standard moves from DIS status to International Standard (IS) status. If, however, the standard does change unexpectedly, the Netman working group will review such changes for appropriate action.

Another rationale for the CMOT approach is that it will facilitate the early use of ISO network management standards in large operational networks. This will make it possible for the Internet community to make valuable recommendations to ISO in the language of OSI management based on actual experience with the use and implementation of these standards. There is continuing network management standards development work in ISO where such contributions would be valuable.

The CMOT architecture is based on the Open Systems Interconnection (OSI) management framework and models developed by ISO. This memo contains a set of protocol agreements for implementing a network management system based on this architecture. The protocol agreement sections of this memo must be read in conjunction with ISO and Internet documents defining specific protocol standards. Documents defining the following ISO standards are required for the implementor: Abstract Syntax Notation One (ASN.1) [5, 6], Association Control (ACSE) [7, 8], Remote Operations (ROSE) [9, 10], Common Management Information Services (CMIS) [11], and Common Management Information Protocol (CMIP) [12]. RFC 1085 [13] is required for the specification of a lightweight presentation layer protocol used in this profile. In addition, RFC 1065 [2] and RFC 1066 [3] are required for a definition of the initial SMI and MIB to be used with the CMOT management system.

This memo is divided into two main parts. The first part presents concepts and models; the second part contains the protocol agreements necessary for implementation of the CMOT network management system. The first part of the memo is divided into three sections: section 3

contains tutorial information on the OSI management framework; section 4 defines the basic CMOT approach; and section 5 discusses the area of management information and specifies how the abstract management information defined in the Internet-standard SMI and MIB map into CMIP. The second part of this memo is divided into sections for each of the protocols for which implementors' agreements are needed: CMISE, ACSE, ROSE, and the lightweight presentation protocol. The protocol profile defined in this part draws on the technical work of the OSI Network Management Forum [14] and the Network Management Special Interest Group (NMSIG) of the National Institute of Standards and Technology (NIST) (formerly the National Bureau of Standards). Wherever possible, an attempt has been made to remain consistent with the protocol agreements reached by these groups.

Part I: Concepts and Models

3. The OSI Management Framework

The OSI management framework [15] presents the basic concepts and models required for developing network management standards. OSI management provides the ability to monitor and control network resources, which are represented as "managed objects." The following elements are essential for the description of a network management architecture and the standardization of a network management system: a model or set of models for understanding management; a common structure of management information for registering, identifying, and defining managed objects; detailed specifications of the managed objects; and a set of services and related protocols for performing remote management operations.

3.1. Architectural Overview

The basic concepts underlying OSI network management are quite simple [16]. There reside application processes called "managers" on managing systems (or management stations). There reside application processes called "agents" on managed systems (or network elements being managed). Network management occurs when managers and agents conspire (via protocols and a shared conceptual schema) to exchange monitoring and control information useful to the management of a network and its components. The terms "manager" and "agent" are also used in a loose and popular sense to refer to the managing and managed system, respectively.

The shared conceptual schema mentioned above is a priori knowledge about "managed objects" concerning which information is exchanged. Managed objects are system and networking resources (e.g., a modem, a protocol entity, an IP routing table, a TCP connection) that are subject to management. Management activities are effected through the manipulation of managed objects in the managed systems. Using the management services and protocol, the manager can direct the agent to perform an operation on a managed object for which it is responsible. Such operations might be to return certain values associated with a managed object (read a variable), to change certain values associated with a managed object (set a variable), or perform an action (such as self-test) on the managed object. In addition, the agent may also forward notifications generated asynchronously by managed objects to the manager (events or traps).

The terms "manager" and "agent" are used to denote the asymmetric relationship between management application processes in which the manager plays the superior role and the agent plays the subordinate.

However, the specification of the management protocol (CMIP) defines a peer protocol relationship that makes no assumptions concerning which end opens or closes a connection, or the direction of management data transfer. The protocol mechanisms provided are fully symmetric between the manager and the agent; CMIS operations can originate at either the manager or agent, as far as the protocol is concerned. This allows the possibility of symmetric as well as asymmetric relationships between management processes. Most devices will contain management applications that can only assume the agent role. Applications on managing systems, however, may well be able to play both roles at the same time. This makes possible "manager to manager" communication and the ability of one manager to manage another.

3.2. Management Models

Network management may be modeled in different ways. Three models are typically used to describe OSI management [17, 18]. An organizational model describes ways in which management can be administratively distributed. The functional model describes the management functions and their relationships. The information model provides guidelines for describing managed objects and their associated management information.

3.2.1. The Organizational Model

The organizational model introduces the concept of a management "domain." A domain is an administrative partition of a network or internet for the purpose of network management. Domains may be useful for reasons of scale, security, or administrative autonomy. Each domain may have one or more managers monitoring and controlling agents in that domain. In addition, both managers and agents may belong to more than one management domain. Domains allow the construction of both strict hierarchical and fully cooperative and distributed network management systems.

3.2.2. The Functional Model

The OSI Management Framework [15] defines five facilities or functional areas to meet specific management needs. This has proved to be a helpful way of partitioning the network management problem from an application point of view. These facilities have come to be known as the Specific Management Functional Areas (SMFAs): fault management, configuration management, performance management, accounting management, and security management. Fault management provides the ability to detect, isolate, and correct network problems. Configuration management enables network managers to change the configuration of remote network elements. Performance

management provides the facilities to monitor and evaluate the performance of the network. Accounting management makes it possible to charge users for network resources used and to limit the use of those resources. Finally, security management is concerned with managing access control, authentication, encryption, key management, and so on.

3.2.3. The Information Model

The OSI Management Framework considers all information relevant to network management to reside in a Management Information Base (MIB), which is a "conceptual repository of management information." Information within a system that can be referenced by the management protocol (CMIP) is considered to be part of the MIB. Conventions for describing and uniquely identifying the MIB information allow specific MIB information to be referenced and operated on by the management protocol. These conventions are called the Structure of Management Information (SMI). The information model is described more fully in section 5.

3.3. ISO Application Protocols

The following ISO application services and protocols are necessary for doing network management using the OSI framework: ACSE, ROSE, and CMIS/CMIP. All three of these protocols are defined using ASN.1 [5]. The ASN.1 modules defining each of these protocols are found in the relevant standards documents. The encoding rules for ASN.1 [6] provide a machine-independent network representation for data.

A brief overview of the terminology associated with the OSI application layer structure is presented here. A complete treatment of the subject can be found in the OSI Application Layer Structure document [22].

In the OSI environment, communication between "application processes" is modeled by communication between application entities. An "application entity" represents the communication functions of an application process. There may be multiple sets of OSI communication functions in an application process, so a single application process may be represented by multiple application entities. However, each application entity represents a single application process. An application entity contains a set of communication capabilities called "application service elements." An application service element is a coherent set of integrated functions. These application service elements may be used independently or in combination. Examples of application service elements are X.400, FTAM, ACSE, ROSE, and CMISE.

When communication is required between two application entities, one

or more "application associations" are established between them. Such an association can be viewed as a connection at the level of the application layer. An "application context" defines the set of application service elements which may be invoked by the user of an application association. The application context may prescribe one or more application service elements.

Generally, an "application layer protocol" is realized by the use of the functionality of a number of application service elements. This functionality is provided by the specification of a set of application protocol data units (APDUs) and the procedures governing their use. In general, the operation of an application layer protocol may require the combination of APDUs from different application service elements. The application entity makes direct use of presentation context identifiers for the specification and identification of APDUs.

3.3.1. ACSE

The Association Control Service Element (ACSE) is used to establish and release associations between application entities. Before any management operations can be performed using CMIP, it is necessary for the two application entities involved to form an association. Either the manager or the agent can initiate association establishment. ACSE allows the manager and agent to exchange application entity titles for the purpose of identification and application context names to establish an application context. As stated above, an application context defines what service elements (for instance, ROSE and CMISE) may be used over the association. After the association is established, ACSE is not used again until the association is released by the manager or agent.

3.3.2. ROSE

The Remote Operation Service Element (ROSE) is the ISO equivalent of remote procedure call. ROSE allows the invocation of an operation to be performed on a remote system. The Remote Operation protocol contains an invoke identifier for correlating requests and responses, an operation code, and an argument field for parameters specific to the operation. ROSE can only be invoked once an application association has been established. CMIP uses the transaction-oriented services provided by ROSE for all its requests and responses. CMIP also uses the error response facilities provided by ROSE.

3.3.3. CMISE

The Common Management Information Service Element (CMISE) is the service element that provides the basic management services. The

CMISE is a user of both ROSE and ACSE. The CMISE provides both confirmed and unconfirmed services for reporting events and retrieving and manipulating management data. These services are used by manager and agent application entities to exchange management information. Table 1 provides a list of the CMISE services. In addition, the CMISE also provides the ability to issue a series of (multiple) linked replies in response to a single request.

Service	Type
M-INITIALISE	confirmed
M-TERMINATE	confirmed
M-ABORT	non-confirmed
M-EVENT-REPORT	confirmed/non-confirmed
M-GET	confirmed
M-SET	confirmed/non-confirmed
M-ACTION	confirmed/non-confirmed
M-CREATE	confirmed
M-DELETE	confirmed

Table 1. CMISE Service Summary

CMIS services can be divided into two main classes: management association services and information transfer services. Furthermore, there are two types of information transfer services: management notification services and management operation services. In addition to the other CMIS services, the CMISE provides facilities that enable multiple responses to confirmed operations to be linked to the operation by the use of a linked identification parameter.

3.3.3.1. Management Association Services

CMIS provides services for the establishment and release of application associations. These services control the establishment and normal and abnormal release of a management association. These services are simply pass-throughs to ACSE.

The M-INITIALISE service is invoked by a CMISE-service-user to establish an association with a remote CMISE-service-user for the purpose of exchanging management information. A reply is expected. (A CMISE-service-user is that part of an application process that makes use of the CMISE.)

The M-TERMINATE service is invoked by a CMISE-service-user to release

an association with a remote CMISE-service-user in an orderly manner. A reply is expected.

The M-ABORT service is invoked by a CMISE-service-user or a CMISE-service-provider to release an association with a remote CMISE-service-user in an abrupt manner.

3.3.3.2. Management Notification Services

The definition of notification and the consequent behavior of the communicating entities is dependent upon the specification of the managed object which generated the notification and is outside the scope of CMIS. CMIS provides the following service to convey management information applicable to notifications.

The M-EVENT-REPORT service is invoked by a CMISE-service-user to report an event about a managed object to a remote CMISE-service-user. The service may be requested in a confirmed or a non-confirmed mode. In the confirmed mode, a reply is expected.

3.3.3.3. Management Operation Services

The definition of the operation and the consequent behavior of the communicating entities is dependent upon the specification of the managed object at which the operation is directed and is outside the scope of CMIS. However, certain operations are used frequently within the scope of management and CMIS provides the following definitions of the common services that may be used to convey management information applicable to the operations.

The M-GET service is invoked by a CMISE-service-user to request the retrieval of management information from a remote CMISE-service-user. The service may only be requested in a confirmed mode. A reply is expected.

The M-SET service is invoked by a CMISE-service-user to request the modification of management information by a remote CMISE-service-user. The service may be requested in a confirmed or a non-confirmed mode. In the confirmed mode, a reply is expected.

The M-ACTION service is invoked by a CMISE-service-user to request a remote CMISE-service-user to perform an action. The service may be requested in a confirmed or a non-confirmed mode. In the confirmed mode, a reply is expected.

The M-CREATE service is invoked by a CMISE-service-user to request a remote CMISE-service-user to create another instance of a managed object. The service may only be requested in a confirmed mode. A

reply is expected.

The M-DELETE service is invoked by a CMISE-service-user to request a remote CMISE-service-user to delete an instance of a managed object. The service may only be requested in a confirmed mode. A reply is expected.

4. The CMOT Architecture

The CMOT (CMIP Over TCP/IP) architecture is based on the OSI management framework [15] and the models, services, and protocols developed by ISO for network management. The CMOT architecture demonstrates how the OSI management framework can be applied to a TCP/IP environment and used to manage objects in a TCP/IP network. The use of ISO protocols for the management of widely deployed TCP/IP networks will facilitate the ultimate migration from TCP/IP to ISO protocols. The concept of proxy management is introduced as a useful extension to the architecture. Proxy management provides the ability to manage network elements that either are not addressable by means of an Internet address or use a network management protocol other than CMIP.

The CMOT architecture specifies all the essential components of a network management architecture. The OSI management framework and models are used as the foundation for network management. A protocol-dependent interpretation of the Internet SMI [2] is used for defining management information. The Internet MIB [3] provides an initial list of managed objects. Finally, a means is defined for using ISO management services and protocols on top of TCP/IP transport protocols. Management applications themselves are not included within the scope of the CMOT architecture. What is currently standardized in this architecture is the minimum required for building an interoperable multivendor network management system. Applications are explicitly left as a competitive issue for network developers and providers.

4.1. Management Models

The following sections indicate how the CMOT architecture applies the OSI managements models and point out any limitations the CMOT architecture has as it is currently defined in this memo.

4.1.1. The Organizational Model

It is beyond the scope of this memo to define the relations and interactions between different management domains. The current CMOT architecture concerns itself only with the operations and characteristics of a single domain of management. The extension of

the mechanisms defined here to include multiple domains is left for further study.

4.1.2. The Functional Model

The CMOT architecture provides the foundation for carrying out management in the five functional areas (fault, configuration, performance, accounting, and security), but does not address specifically how any of these types of management are accomplished. It is anticipated that most functional requirements can be satisfied by CMIS. The greatest impact of the functional requirements in the various areas will likely be on the definition of managed objects.

4.1.3. The Information Model

There are two different SMI specifications that are important to the CMOT architecture. The first is the SMI currently being defined by ISO [19]. This SMI is important to the CMOT approach because the ISO management protocol CMIP has been designed with the ISO model of management information in mind. The second SMI of importance is the that defined by the IETF MIB working group for use in defining the Internet MIB [3]. This Internet SMI, which is loosely based on a simplified version of the ISO SMI, is important because the managed objects defined for TCP/IP networks to be used by CMOT are defined in terms of it. Thus, in order to make the CMOT architecture complete, it will be necessary to show how the Internet SMI maps into CMIP in such a way as to enable it to convey the management information defined in the Internet MIB. This is done in the section devoted to management information (section 5).

4.2. Protocol Architecture

The objective of the CMOT protocol architecture is to map the OSI management protocol architecture into the TCP/IP environment. The model presented here follows the OSI model at the application layer, while using Internet protocols at the transport layer. The ISO application protocols used for network management are ACSE, ROSE, and CMIP. Instead of implementing these protocols on top of the ISO presentation, session, and transport layer protocols, the protocol data units (PDUs) for ACSE, ROSE, and CMIP are carried using the Internet transport protocols UDP [20] and TCP [21]. This is made possible by means of the lightweight presentation protocol defined in RFC 1085 [13] that maps ROSE and ACSE onto TCP/UDP/IP. The use of Internet transport protocols is transparent to network management applications, since they are presented with real ISO services.

4.2.1. The Lightweight Presentation Layer

Given that it is desired to put ISO application protocols on top of TCP/IP, how is this best accomplished? It is necessary somehow to fill the "gap" between the ISO protocols (ACSE and ROSE) and the Internet protocols (UDP and TCP). Two basic approaches were considered.

One possible approach [23] is to extend the ISO portion of the protocol stack down to the transport layer. The ISO Transport Protocol Class 0 (TP 0) then uses TCP instead of an ISO network protocol. Effectively, this treats TCP as a reliable network connection analogous to X.25. This approach allows us to operate "standard" ISO applications over TCP regardless of their service requirements, since all ISO services are provided. In this case, network management is just another such application. The major drawback with this approach is that full ISO presentation, session, and transport layers are expensive to implement (both in terms of processing time and memory).

Another approach is presented in RFC 1085. Since the service elements required for network management (ACSE, ROSE, CMISE) do not require the use of full ISO presentation layer services, it is possible to define a "streamlined" presentation layer that provides only the services required. This lightweight presentation protocol (LPP) allows the use of ISO presentation services over both TCP and UDP. This approach eliminates the necessity of implementing ISO presentation, session, and transport protocols for the sake of doing ISO network management in a TCP/IP environment. This minimal approach is justified because this non-ISO presentation protocol used is very small and very simple. Thus, the LPP defined in RFC 1085 provides a compact and easy to implement solution to the problem. The resulting CMOT protocol stack is shown in Figure 1.

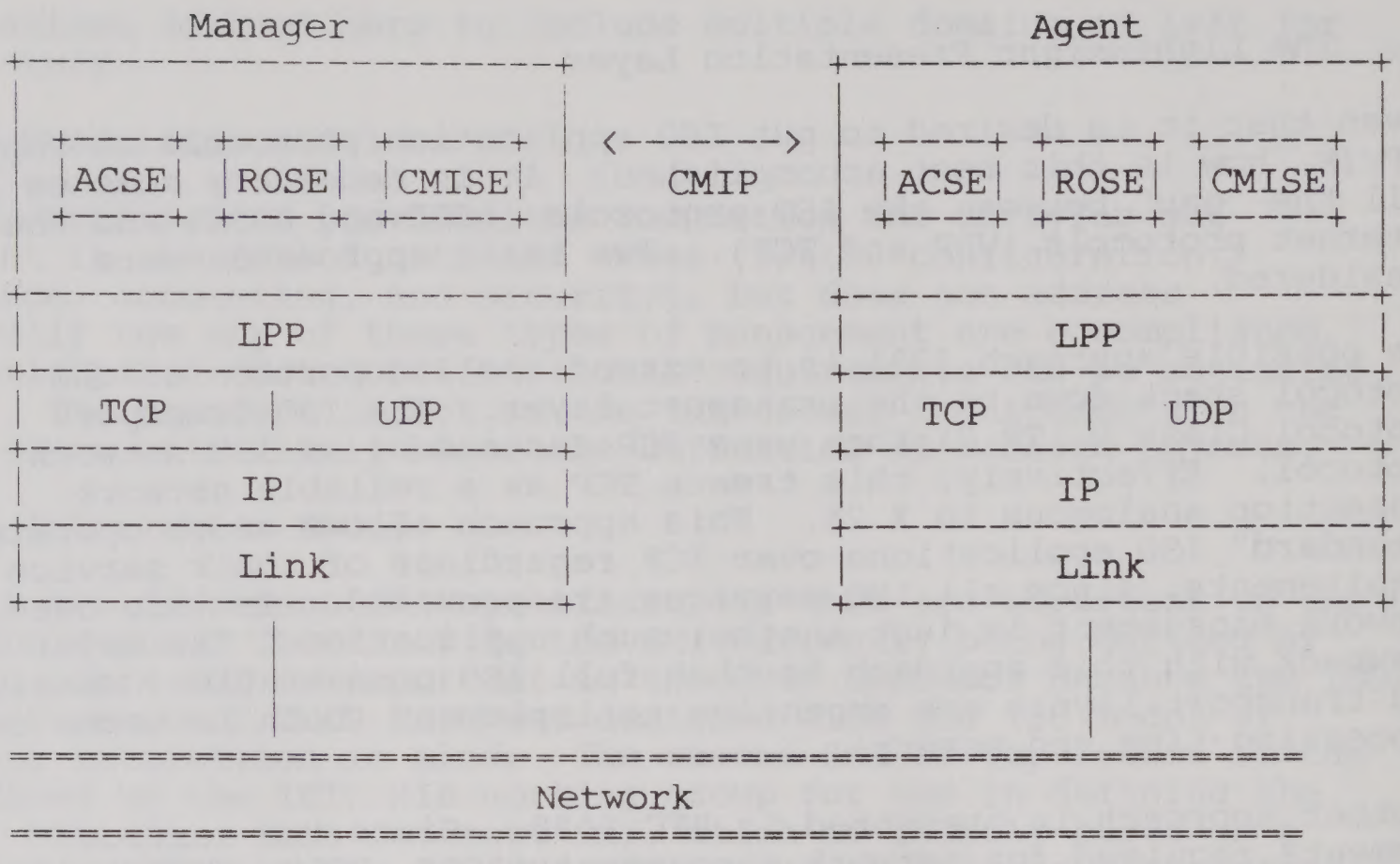

Figure 1. The CMOT Protocol Architecture

It is important to note that the presentation services provided by the LPP are "real" (but minimal) ISO presentation services [24]. This provides a clear migration path to "full ISO" in the future. Such a migration would be accomplished by substituting ISO protocols for the Internet protocols TCP, UDP, and IP [25], and replacing the LPP with ISO presentation and session protocols. No changes will be required in the ISO application layer protocols. For this reason, investments in application development will be well preserved.

4.2.2. The Quality of Transport Service

The quality of transport service needed for network management applications is an issue that has caused much controversy, yet it has never been resolved. There are two basic approaches: datagram-oriented and connection-oriented. There are advantages and disadvantages to both of these two approaches. While the datagram-oriented approach is simple, requires minimal code space, and can operate under conditions where connections may not be possible, the connection-oriented approach offers data reliability and provides guaranteed and consistent service to the driving application.

This memo does not take sides on this issue. Rather it passes such

resolution to the network management applications, which are ultimately the point where the requirements from the underlying service need to be determined. As such, the CMOT protocol architecture provides both services. The presentation layer service allows the application to select either high or low quality service for the underlying transport. Depending on this choice, the LPP will use either UDP (low quality) or TCP (high quality) to establish the application association and carry the application data. It is important, however, for the application to be aware of the quality of service that it is using: low quality means low quality! The use of an unreliable transport like UDP necessarily puts more burden on the application.

4.3. Proxy Management

Proxy is a term that originated in the legal community to indicate an entity empowered to perform actions on behalf of another. In our context, a proxy is a manager empowered to perform actions on behalf of another manager. This may be necessary because the manager cannot communicate directly with the managed devices either for security or other administrative reasons or because of incompatible communication mechanisms or protocols. In either case, the proxy assumes the agent role with respect to the requesting manager and the manager role with respect to the managed device.

Some network elements, such as modems or bridges, may not be able to support CMIP and all the associated protocols. In addition, such devices may not have Internet addresses. Such devices are called "limited systems". It may be possible to manage these devices using proprietary mechanisms or other standard protocols (such as the IEEE 802.1 management protocol for managing bridges). In cases where it is desirable to integrate the management of such devices with the overall CMOT management of an internet, it is necessary to use proxy management. Some network elements that are not "limited systems" as described above may still benefit from the use of proxy management. If the management protocol supported by such a system is proprietary or some standard protocol other than CMIP (such as SNMP), then CMOT proxy management can be used to integrate the management of such systems.

A proxy operates in the following manner. When a CMOT manager wants to send a request to a managed device that it cannot communicate with directly, it routes the request to the proxy. The proxy maps the CMIP request into the information schema understood by the managed device and sends the appropriate request to the managed device using the native management protocol of the device. When the proxy receives the response from the managed device, it uses CMIP to return the information to the manager that made the original request.

The use of proxy management can be largely transparent to the requesting manager, which appears to be exchanging information directly with the selected device. The only thing that is known to the manager is that additional "instance" information is required to select a particular device managed by the proxy. Each proxy may support many managed devices, using the "instance" information to multiplex CMIP requests and responses among them. The mapping between a specific instance and an actual managed device is a local matter. (The use of the CMIP Object Instance field to select a particular system to manage by proxy is explained below in section 5.3.2.2.)

A proxy may also serve as an "intermediate manager" in another less transparent sense. The proxy manager may be requested to calculate summary statistics on information gathered from many different managed systems (e.g., the average number of PDUs transmitted or the distribution of PDUs transmitted over time). The proxy may be requested to log events transmitted by the managed systems under its control and to send to the requesting manager only those events of specific types. When this use of proxy management is made, the conceptual schema for managed objects known to both the requesting manager and proxy must include definitions of these aggregate managed objects (i.e., objects that do not belong to any one managed system). How the aggregate statistics would be calculated and logging performed based on information from the different devices managed by the proxy would be part of the definition of these aggregate managed objects.

4.4. Directory Service

RFC 1085 specifies the use of a minimal (or "stub") directory service. It specifies how the service name for an OSI application entity is converted into an "application entity title." The application entity title is then mapped into a presentation address. The form of a service name, an application entity title, and a presentation address can be found in RFC 1085.

5. Management Information

The description of management information has two aspects. First, a structure of management information (SMI) defines the logical structure of management information and how it is identified and described. Second, the management information base (MIB), which is specified using the SMI, defines the actual objects to be managed. The purpose of this section is to show how CMIP is used in the CMOT architecture to convey information defined in the Internet MIB.

5.1. The Structure of Management Information

The SMI supplies the model for understanding management information, as well as templates and ASN.1 macros that can be used for defining actual management information. The following sections discuss the ISO SMI, the Internet SMI, and a way of interpreting the Internet SMI in terms of the ISO SMI so that CMIP can be used to carry management information defined in terms of the Internet SMI.

5.1.1. The ISO SMI

The ISO SMI [19] is based on the abstraction of a "managed object" and the various kinds of relationships objects can be involved in. The following discussion does not purport to be a complete and accurate description of the latest ISO SMI work. It is intended to be a clear presentation of the basic ISO SMI concepts essential for understanding the CMIP-specific interpretation of the Internet SMI presented in section 5.3.

5.1.1.1. Managed Objects and Attributes

Management Information is modeled using object-oriented techniques. All "things" in the network that are to be managed are represented in terms of managed objects. A "managed object" is an abstraction (or logical view) for the purposes of network management of a "manageable" physical or logical resource of the network. In this context, "manageable" means that a particular resource can be managed by using CMIP. Examples of managed objects are protocol entities, modems, and connections.

Each managed object belongs to a particular object class. An "object class" represents a collection of managed objects with the same, or similar, properties. A particular managed object existing in a particular network is defined as an "object instance" of the object class to which it belongs. Thus, an object instance represents an actual realization of an object class (i.e., a managed object of a particular class bound to specific values). An example of an object class is "transport connection." In an actual network, there are a number of managed objects (specific transport connections) that are instances of this class. In summary, a managed object type, which is called an "object class," is the collection of all actual and potential instances of that type.

Managed objects are fully defined by specifying the "attributes" or properties the object has, the CMIS operations that can be performed on the object (e.g., M-SET, M-CREATE) and any constraints on those operations, specific actions (e.g., self-test) that can be performed on the object, events that the object can generate, and information

about various relationships the object may be involved in. All of this information relevant to a managed object is typically provided by filling in an object template.

Managed objects contain properties that are referred to as attributes. Attributes are atomic items of information that can only be manipulated as a whole. An example of an attribute is a counter providing a specific piece of information, such as the number of packets retransmitted.

Each object class and attribute is assigned a unique identifier (an ASN.1 OBJECT IDENTIFIER) for purposes of naming by a registration authority.

5.1.1.2. Management Information Hierarchies

Managed objects participate in relationships with each other. There are two relationships that are of particular importance for management information: the containment relationship and the inheritance relationship. These relationships can be used to construct hierarchies of managed objects. In addition, there is another hierarchy defined by the registration process for registering identifiers for object classes and attributes.

5.1.1.2.1. The Registration Hierarchy

The registration hierarchy is determined by the ASN.1 registration tree [5] for assigning OBJECT IDENTIFIERs. An OBJECT IDENTIFIER is an administratively assigned name composed of a series of integers traversing a path from the root of the ASN.1 registration tree to the node or leaf to be identified. For example, the sequence of integers { iso(1) standard(0) ips-osi-mips(9596) cmip(2) } (1.0.9596.2) can be used to uniquely identify the CMIP standard. Each node of this tree has an associated registration authority that determines how numbers in the subtree defined by that node are allocated. In the context of management, these OBJECT IDENTIFIERs are used for identifying object classes and attributes. The registration hierarchy is not based on any particular relationship between managed objects or between managed objects and their attributes. It is independent of both the inheritance and containment relationships described below. Its purpose is simply to generate universally unique identifiers.

5.1.1.2.2. The Containment Hierarchy

The containment hierarchy is constructed by applying the relationship "is contained in" to objects and attributes. Objects of one class may contain objects of the same or different class. Objects may also contain attributes. Attributes cannot contain objects or other

attributes. For example, objects of the class "transport entity" may contain objects of the class "transport connection"; an object of the class "management domain" may contain objects of the class "node." An object class that contains another object class is called the "superior" object class; an object class that is contained in another object class is called the "subordinate" object class. The containment relationships that an object may participate in are part of the definition of the object class to which that managed object belongs. All object classes (except the topmost) must have at least one possible superior in the containment tree. The definition of a class may permit it to have more than one such superior. However, individual instances of such a class are nevertheless contained in only one instance of a possible containing class.

The containment hierarchy is important because it can be used for identifying instances of a managed object. For example, assume there is an object class "domain" that contains an object class "node" that contains an object class "transport entity" that contains an object class "transport connection." A particular instance of a transport connection can be identified by the concatenation of "instance information" for each object class in the containment path: { domain="organization," node="herakles," transport entity=tp4, transport connection=<TSAP-AddressA, TSAP-AddressB> }.

What constitutes appropriate "instance information" for each object class is part of the definition of that object class and is known as the "distinguished attribute(s)." A distinguished attribute is composed of an OBJECT IDENTIFIER naming the attribute and the value of the attribute. For each object class, the distinguished attributes that differentiate instances of that class are collectively called the "relative distinguished name." A sequence of relative distinguished names (one for each class in the containment path) is the "distinguished name" of a managed object. The example given above represents the distinguished name of a transport connection. The containment hierarchy is sometimes referred to as the "naming tree", because it is used to "name" a particular instance of a managed object.

The containment relationship also defines an existence dependency among its components; an object or attribute can "exist" only if the containing object also "exists." Deletion of an object may result in deletion of all objects and attributes contained within it. Alternately, depending on the definition of the managed object, deletion may be refused until all contained managed objects have been deleted.

5.1.1.2.3. The Inheritance Hierarchy

The inheritance hierarchy is constructed by applying the relationship "inherits properties of" to object classes. An object class may inherit properties of another object class; refinement is obtained by adding additional properties. In this relationship, the parent class is called the "superclass" and the inheriting class the "subclass." For example, the class "layer entity" may be a superclass of "network entity," which in turn is a superclass of "X.25 network entity." Attributes defined for "network entity" (e.g., the number of packets sent) are automatically defined for "X.25 network entity" without having to explicitly include them in the definition for the class "X.25 network entity." Thus, inheritance serves as a shorthand for defining object classes using object-oriented methodology. Each class (except the topmost) has at least one superclass, but may have zero, one, or many subclasses. Subclasses may in turn have further subclasses, to any degree. A special object called "top" is the ultimate superclass. It has no properties of its own.

The inheritance hierarchy has no relevance to the naming of object instances. It is useful only insofar as it leads to a manageable and extensible technique for the definition of object classes.

5.1.2. The Internet SMI

The Internet SMI [2] is designed to be a protocol-independent SMI that can be used with both SNMP and CMIP. For this reason, it is necessary for any management protocol that uses this SMI to show how it is to be interpreted in a protocol-specific manner. This is done for CMIP in this memo.

The Internet SMI indicates both how to identify managed objects and how to define them. The Internet SMI defines a registration subtree rooted at { iso(1) org(3) dod(6) internet(1) } for the sake of registering OBJECT IDENTIFIERs to be used for uniquely identifying managed objects. The current Internet SMI specifies the format for defining objects in terms of an "object type" template and an associated OBJECT-TYPE ASN.1 macro. An object type definition contains five fields: a textual name, along with its corresponding OBJECT IDENTIFIER; an ASN.1 syntax; a definition of the semantics of the object type; an access (read-only, read-write, write-only, or not-accessible); and a status (mandatory, optional, or obsolete). The current Internet SMI does not provide any mechanism for defining actions or events associated with a managed object.

In describing management information, the current Internet SMI does not use the notions of "object class" and "attribute" found in the ISO SMI. Only the concepts of "object type" and "object instance"

are used. The Internet SMI shows how to define object types; it leaves the specification of object instances as a protocol-specific matter. The current Internet structure of management information is simpler and less rich than the corresponding ISO structure. The ISO SMI makes a distinction between simple "attributes," which can be viewed as "leaf objects" that are the lowest elements of the containment hierarchy, and composite "managed objects" that belong to an "object class" and have a structure associated with them (that is, can contain attributes). The Internet SMI does not draw this distinction; both simple and composite "objects" are defined as "object types." What structure is associated with objects in the Internet SMI is defined through the deliberate attempt to structure the lower part of the Internet registration tree according to containment principles. (Objects that are considered "attributes" of other containing objects are defined directly below them in the object registration tree.) This results in a certain lack of flexibility, since the registration hierarchy is implicitly used to define the containment hierarchy. This means that the Internet SMI does not contain a mechanism for defining containment relationships that do not happen to coincide with the registration hierarchy. In interpreting the Internet SMI for use with CMIP, it is necessary to overcome this limitation.

5.2. The Management Information Base

The Management Information Base (MIB) is a "conceptual repository of management information." It is an abstract view of all the objects in the network that can be managed. Note that the MIB is conceptual in that it does not carry any implications whatsoever about the physical storage (main memory, files, databases, etc.) of management information. The SMI provides the guidelines for defining objects contained in the MIB.

The CMOT approach will use the Internet MIB based on the Internet SMI described above. The first version of the Internet MIB, which is the product of the IETF MIB working group, is defined in RFC 1066 [3]. It contains objects divided into eight groups: system, interfaces, address translation, IP, ICMP, TCP, UDP, and EGP. In addition, the Internet SMI provides for future versions of the Internet MIB and a means for otherwise extending the MIB through the registration of managed objects under "private" and "experimental" branches of the object registration tree. Appendix B provides a protocol-specific interpretation of the first version of the TCP/IP MIB defined in [3] so that it can be used with CMOT. This interpretation is based on a straightforward mapping of the current Internet SMI to the ISO SMI (section 5.3).

The initial version of the Internet MIB concentrates on defining

objects associated with various Internet protocols. It is expected that future versions of the Internet MIB and various extensions will provide a much richer set of objects to manage, including management information about a variety of network devices and systems. Thus, an expanded MIB will allow wide-ranging and powerful management using the CMOT approach.

5.3. An Interpretation of the Internet SMI

In order to use CMIP to convey information defined in terms of the Internet SMI, it is necessary to show how object instances are specified and to provide the necessary structure for differentiating object class and attributes. These objectives are both met by separating the containment hierarchy used for naming objects from the registration hierarchy and by imposing an "object class" structure on the Internet SMI. Using the technique of imposing an object class structure does not replace or redefine the object definitions in the Internet MIB; it merely provides a necessary gloss or commentary on a MIB defined in terms of the Internet SMI. For example, Appendix B references the "object type" definitions found in [3], but imposes additional structure on them.

This object class definition derives from a simplified version of the OBJECT-CLASS macro defined in the ISO SMI [19]. The more complex definition is not needed for present purposes. (The object class definition presented here could be extended in the future to show what actions and events are associated with a managed object.) The object class definition has the following fields:

OBJECT CLASS:

A textual name, termed the OBJECT CLASS DESCRIPTOR, for the object class, along with its corresponding OBJECT IDENTIFIER.

Definition:
A textual description of the object class.

Subclass Of:
The OBJECT CLASS DESCRIPTOR of the object class that is the superclass of this object class. This field is used for indicating the inheritance relationship.

Superiors:
A list of OBJECT CLASS DESCRIPTORs of the possible superior object classes of this object class. This field is used for indicating the containment relationship.

Names:
A list of OBJECT DESCRIPTORs identifying the OBJECT TYPES that are the distinguished attributes of this object class. (The OBJECT-TYPE macro is defined in RFC 1065). Attributes listed here will normally be present in the Attribute field of the object class definition. This field is used for indicating what attributes must be present in the relative distinguished name that indicates an instance of this object class.

Attributes:
A list of OBJECT DESCRIPTORs identifying the OBJECT TYPES that are attributes of this object class. (The OBJECT-TYPE macro is defined in RFC 1065). This field is used for indicating the attributes that are contained in this object class.

This object class definition satisfies our objectives for interpreting the Internet SMI for use by CMIP. The Attributes field shows what attributes are contained in this object class; this makes the necessary distinction between object classes and attributes required by CMIP. Instead of referencing an "attribute" def inition (as is done in the ISO SMI), the Attributes field references the "object type" definition found in RFC 1065 and used to define the Internet-standard MIB in RFC 1066. The name, syntax, and access information required for attributes is contained in the "object type" definition. Two things are required for specifying an instance of a managed object: a containment relationship determining a sequence of object classes and a means for specifying the distinguished attributes for an object class. The Superiors field makes the containment relationship explicit; it is no longer merely a function of the registration tree. The Names field makes it possible to indicate the distinguished attributes for an object class required for giving instance information. Thus, the object class definition makes it possible to specify an object instance using CMIP.

5.3.1. Object Class and Attributes

The mapping of management information to the CMIS parameters Managed Object Class and Attribute Identifier List now becomes apparent.

5.3.1.1. Object Class

The CMIS Managed Object Class parameter is the OBJECT IDENTIFIER assigned to the particular object class. For example, the Managed Object Class for the object class "ip" (as defined in Appendix B) is

{ mib 4 } = 1.3.6.1.2.1.4.

5.3.1.2. Attribute Identifier

The CMIS Attribute Identifier List parameter is a list of Attribute Identifiers. An Attribute Identifier can be either global or local. If it is global, then it is the OBJECT IDENTIFIER assigned to the attribute (i.e., "object type") that is being indicated. For example, the global Attribute Identifier for the attribute "ipForwarding" (as defined in [3]) is

{ ip 1 } = 1.3.6.1.2.1.4.1.

If the Attribute Identifier is local, it is an integer that is the last component in the OBJECT IDENTIFIER identifying the object. For ipForwarding, the local Attribute Identifier is 1. In the case where the local identifier is used, the leading components of the OBJECT IDENTIFIER for the attribute must be the OBJECT IDENTIFIER of the containing object class. This is true for the interpreted Internet MIB defined in Appendix B, but may not be true generally. The local identifier is intended to be interpreted relative to the Managed Object Class field of the CMIP PDU. When a local Attribute Identifier is encountered in a CMIP PDU, the global form of the identifier is formed by prepending the OBJECT IDENTIFIER in the Managed Object Class field to the local identifier. This is valid only when scoping is not used (i.e., scoping is "baseObject"). If scoping is used, then the global form of the Attribute Identifier must be used instead of the local form.

5.3.2. Management Information Hierarchies

The following sections show how the three management information hierarchies are to be understood for the interpreted Internet SMI.

5.3.2.1. The Registration Hierarchy

The registration hierarchy is the global object registration tree described in [2]. It is used merely for assigning identifiers for object classes and attributes (i.e., "object types" in RFC 1065).

5.3.2.2. The Containment Hierarchy

As described above, the containment hierarchy is used to specify an object instance. The Names field of the object class definition contains the distinguished attributes for the object class. The OBJECT IDENTIFIER naming the "attribute" together with its value is called an attribute value assertion. A set of attribute value assertions (one for each distinguished attribute) is the relative distinguished name associated with that object class. The sequence of relative distinguished names for each of the object classes in the

containment hierarchy to which a managed object belongs is the distinguished name of the object. An object instance is fully specified by a distinguished name.

Let us take a concrete example from Appendix B. How would we represent an instance of an entry in the IP routing table? We begin by examining the object class in question (ipRouteEntry) and use the Superiors field to find the superior class in the containment hierarchy (ipRoutingTable). This process continues until we construct the following containment path of object classes: system, ip, ipRoutingTable, ipRouteEntry. Now for each of these object classes, we inspect the Names field to find the distinguished attribute for that object class. If no Names field is present (as is the case for "ip" and "ipRoutingTable"), then no instance information is required at that level. Both "system" and "ipRouteEntry" have Name fields to show what information is expected at that level. With this information, we can construct the following distinguished name specifying an instance of an IP routing table entry:

```
baseManagedObjectInstance {
   distinguishedName {
      relativeDistinguishedName {    -- system
         attributeValueAssertion {
            attributeType { cmotSystemID }
            attributeValue "gateway1.acme.com"
         }
      },
      relativeDistinguishedName {    -- ipRouteEntry
         attributeValueAssertion {
            attributeType { ipRouteDest }
            attributeValue 10.0.0.51
         }
      }
   }
}
```

If the system instance information is not present, then it is assumed to be the system with which the management association is established (i.e., the system receiving the request).

Note that the object instance tree can contain components of the distinguished name that are outside the managed system (node). This enables referencing of objects across management domains (there could be an object class "domain") and across a collection of nodes. In a network where several intermediate managers may be involved in a request, each intermediate manager can use the "system" portion of

the name to determine where to send a request or result. This technique of naming treats each intermediate managing system as a proxy manager. The proxy manager resolves the address of the next node in the chain and may use a different protocol to transfer the request or result. Thus, the "system" instance information can be used to name devices being managed by proxy.

5.3.2.3. The Inheritance Hierarchy

The Internet SMI does not use the inheritance relationship. The "Subclass Of" field is present in the object class definition to show how the inheritance relationship would be represented and to allow for future extensibility. It is not used for any of the object classes defined in Appendix B.

5.4. Scoping, Filtering, and Synchronization

Within some services, CMIS provides additional capabilities that are related to the SMI. These are the scoping, filtering, synchronization, and linked-reply facilities. The presence of these facilities are indicated by the Multiple Object Selection Functional Unit defined in CMIS [11].

These facilities provide the manager with the ability to operate on a collection of managed objects, rather than a single object. The selection of multiple objects occurs in two phases: scoping and filtering. Scoping is used to identify the managed objects to which a filter is to be applied. Then filtering is used to select a subset of managed objects that satisfy certain conditions. If scoping is not used, only the "base" managed object indicated by the CMIS Managed Object Class parameter is implied. An example of the use of scoping and filtering for selecting a particular managed object (a table entry) is given in one of the sample protocol exchanges found in Appendix C.

5.4.1. Scoping

Scoping is meant to be understood in terms of the containment hierarchy. A position at a certain level of the containment tree is defined by the CMIS Managed Object Class parameter. The CMIS Scope parameter is then interpreted relative to this "base" managed object (defined by both object class and object instance). The Scope parameter can be used to select the base object alone, all managed objects in the entire subtree (of the containment tree) below the base object, or all managed objects in the "n"th level (n = 1, 2, 3,...) below the base object.

5.4.2. Filtering

Within the objects selected as a result of the scope parameter, it is possible to further refine the selection of managed objects through the use of filtering. Filtering provides the ability to select a subset of these objects based on conditions applied to attributes (e.g., IP routing table entries with the "ipRouteAge > 100") and logical operations (and, or, not).

5.4.3. Synchronization

When multiple managed objects have been selected using scoping and filtering, the question of synchronization across object instances (such as multiple IP routing table entries) arises. The two possible choices are "best effort" and "atomic." If "best effort" synchronization is selected, the failure to apply an operation (e.g., M-SET) to one instance of an object does not affect the effort to apply this operation to other instances of the object. If "atomic" synchronization is selected, then the operation is either performed on all object instances selected or none. The default synchronization is best effort.

5.4.4. Linked Replies

If the reply to a single request for a set of managed objects results in more than one managed object being returned, all of these managed objects cannot be returned together in a single CMIP response PDU. The reason for this is that the structure of the CMIP response PDU only has a single field for containing object instance information. Since each managed object has its own instance information, each managed object must be returned in a separate CMIP PDU. In such a case, the CMIP Linked Reply PDU is used. The Linked Reply PDU provides a means of associating each of the multiple replies with the original request that generated them. Thus, a single CMIP Get Request PDU that uses scoping and filtering would result in zero or more CMIP Linked Reply PDUs being returned before a final CMIP Get Result PDU.

A linked reply can also be used to segment a CMIP response pertaining to a single managed object. This would only be necessary if UDP is being used as the underlying transport and it is not possible to return all the information requested about the managed object in a single response PDU subject to the size limitations described in section 10.2.

5.5. Accessing Tables

This section explains how to use the interpreted Internet SMI and MIB

to access tables.

5.5.1. Accessing Whole Tables

A whole table is accessed by specifying the object class of the table, indicating a scoping level of one, and not providing an attribute identifier list. The CMIS standard [11] specifies that if the attribute identifier parameter is not present, then all attribute identifiers are assumed. The following CMIS parameters would be used to return the entire TCP connection table:

```
Object Class: { tcpConnTable }
Object Instance: "empty" (unless proxy management is used)
Scope: oneLevel(1)
Filter: not present
Attribute Identifier List: not present
```

By scoping one level below "tcpConnTable," all managed objects of the class "tcpConnEntry" are selected. (The object class "tcpConnEntry" is the only object class one level below the object class "tcpConnTable" in the containment hierarchy.) The absence of an attribute identifier list signals that all attributes of the managed object are to be returned (i.e., all fields of the TCP connection table entry).

In reply to this request, each entry of the table will be returned in a separate CMIP PDU (either a Linked Reply PDU or a Get Result PDU). Each reply CMIP PDU will specify the Object Class "tcpConnEntry" and the appropriate Object Instance information for that entry, as well as an Attribute List giving the values of each of the fields of the table entry.

5.5.2. Accessing Table Entries

An entire table entry is accessed by specifying the object class of the table entry, providing a distinguished name specifying the instance of the table entry, and not providing an attribute identifier list. As seen above, the absence of the attribute identifier list parameter indicates that all attributes are assumed. The absence of a scope parameter indicates that the base managed object class is intended. The following CMIS parameters would be used to return the entire IP routing table entry for which the field "ipRouteDest" has the value 10.0.0.51:

```
Object Class: { ipRouteEntry }
Object Instance: { ipRouteDest, 10.0.0.51 }
Scope: not present
Filter: not present
```

Attribute Identifier List: not present

The result is returned in a single CMIP Get Result PDU with an attribute list consisting of all of the attributes (i.e., fields) of the table entry and their corresponding values.

If the object class field refers to a table entry and no instance information is provided to select a particular entry, then a "noSuchObjectInstance" CMIP error should be returned.

Part II: Protocol Agreements

6. CMOT Protocol Overview

This part of the document is a specification of the protocols of the CMOT architecture. Contained herein are the agreements required to implement interoperable network management systems using these protocols. The protocol suite defined by these implementors' agreements will facilitate communication between equipment of different vendors, suppliers, and networks. This will allow the emergence of powerful multivendor network management based on ISO models and protocols.

The choice of a set of protocol standards together with further agreements needed to implement those standards is commonly referred to as a "profile." The selection policy for the CMOT profile is to use existing standards from the international standards community (ISO and CCITT) and the Internet community. Existing ISO standards and draft standards in the area of OSI network management form the basis of this CMOT profile. Other ISO application layer standards (ROSE and ACSE) are used to support the ISO management protocol (CMIP). To ensure interoperability, certain choices and restrictions are made here concerning various options and parameters provided by these standards. Internet standards are used to provide the underlying network transport. These agreements provide a precise statement of the implementation choices made for implementing ISO network management standards in TCP/IP-based internets.

In addition to the Netman working group, there are at least two other bodies actively engaged in defining profiles for interoperable OSI network management: the National Institute of Science and Technology (NIST) Network Management Special Interest Group (NMSIG) and the OSI Network Management Forum. Both of these groups are similar to the Netman working group in that they are each defining profiles for using ISO standards for network management. Both differ in that they are specifying the use of underlying ISO protocols, while the Netman working group is concerned with using OSI management in TCP/IP networks. In the interest of greater future compatibility, the Netman working group has attempted to make the CMOT profile conform as closely as possible to the ongoing work of these two bodies.

6.1. The CMOT Protocol Suite

The following seven protocols compose the CMOT protocol suite: ISO ACSE, ISO DIS ROSE, ISO DIS CMIP, the lightweight presentation protocol (LPP), UDP, TCP, and IP. The relation of these protocols to each other is briefly summarized in Figure 2.

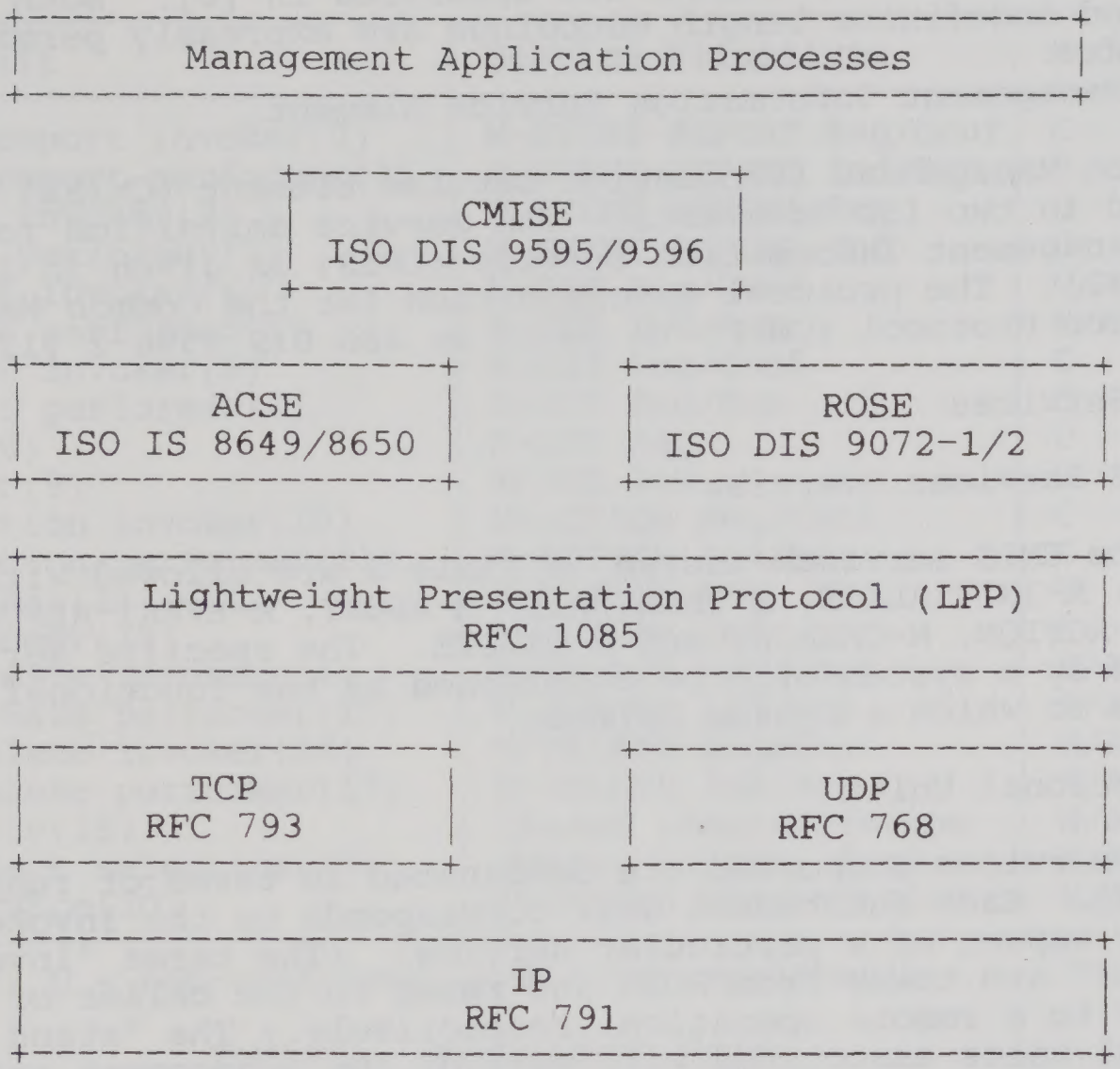

Figure 2. The CMOT Protocol Suite

6.2. Conformance Requirements

A CMOT-conformant system must implement the following protocols: ACSE, ROSE, CMIP, LPP, and IP. A conformant system must support the use of the LPP over either UDP or TCP. The use of the LPP over both UDP and TCP on the same system may be supported. A conformant system need not support all CMIS operations. A conformant system must, however, support at least one of the functional unit groups (indicating a set of supported services) defined in section 7.1.3. The service and protocol selections are described in greater detail in the following sections.

6.3. Abstract Syntax Notation_

The abstract syntax notation for all of the application service elements of the CMOT protocol suite is Abstract Syntax Notation One (ASN.1) [5]. The LPP is also defined using ASN.1. The basic

encoding rules used for ASN.1 are specified in [6]. Both definite-length and indefinite-length encodings are expressly permitted.

7. Common Management Information Service Element

The Common Management Information Service Element (CMISE) is specified in two ISO documents. The service definition for the Common Management Information Service (CMIS) is given in ISO DIS 9595-2 [11]. The protocol specification for the Common Management Information Protocol (CMIP) is found in ISO DIS 9596-2 [12].

7.1. CMIS Services

7.1.1. CMIS Services Overview

All of the CMIS services listed in Table 1 are allowed with the CMOT approach: M-INITIALISE, M-TERMINATE, M-ABORT, M-EVENT-REPORT, M-GET, M-SET, M-ACTION, M-CREATE, and M-DELETE. The specific services supported by a system will be determined by the functional unit group or groups to which a system belongs.

7.1.2. Functional Units

The CMIS services supported are designated in terms of functional units [11]. Each functional unit corresponds to the invoker or performer aspect of a particular service. (The terms "invoker" and "performer" are taken from ROSE and refer to the caller of and responder to a remote operation, respectively.) The "stand alone" functional units associated with each of the management services are given in Table 2 as functional units 0-17. The number following the name of each functional unit in the table is defined by CMIP [12] to identify that particular functional unit. The functional units are used by the CMISE-service-user at the time of association establishment to indicate which services it is willing to support.

Functional Unit	Service Primitives	Mode
conf. event report invoker(0)	M-EVENT-REPORT Req/Conf	C
conf. event report performer(1)	M-EVENT-REPORT Ind/Rsp	C
event report invoker(2)	M-EVENT-REPORT Req	U
event report performer(3)	M-EVENT-REPORT Ind	U
confirmed get invoker(4)	M-GET Req/Conf	N/A
confirmed get performer(5)	M-GET Ind/Rsp	N/A
confirmed set invoker(6)	M-SET Req/Conf	C
confirmed set performer(7)	M-SET Ind/Rsp	C
set invoker(8)	M-SET Req	U
set performer(9)	M-SET Ind	U
confirmed action invoker(10)	M-ACTION Req/Conf	C
confirmed action performer(11)	M-ACTION Ind/Rsp	C
action invoker(12)	M-ACTION Req	U
action performer(13)	M-ACTION Ind	U
confirmed create invoker(14)	M-CREATE Req/Conf	N/A
confirmed create performer(15)	M-CREATE Ind/Rsp	N/A
confirmed delete invoker(16)	M-DELETE Req/Conf	N/A
confirmed delete performer(17)	M-DELETE Ind/Rsp	N/A
multiple reply(18)	Linked Identification	N/A
multiple object selection(19)	Scope, Filter, Sync.	N/A
extended service(20)	Extended Presentation	N/A

C = confirmed, U = non-confirmed, N/A = not applicable

Table 2. Functional Units

In addition to the stand alone functional units, there are three additional functional units. If any of these additional functional units are selected, then at least one of the stand alone functional units must be selected. The multiple reply functional unit makes available the use of the linked identification parameter in the selected stand alone functional units. This makes possible the use of linked reply (multiple CMIP PDU responses to a single request). The multiple object selection functional unit makes available the use of the scope, filter, and synchronization parameters in the selected stand alone functional units. If the multiple object selection functional unit is selected, then the multiple reply functional unit must also be selected. The extended services functional unit makes available presentation layer services in addition to the P-DATA service. Selecting this functional unit has no effect in the context of CMOT, since the lightweight presentation layer provides only minimal ISO presentation services.

7.1.3. Functional Unit Groups

In order to assist in the reduction of code size and complexity for different types of devices, a number of "functional unit groups" have been defined. Each of these groups indicates a set of services defined for either a manager or an agent. The "negotiation" concerning which functional unit groups are supported is done by means of the Functional Units parameter of the M-INITIALISE service (see section 7.1.4.1). There are five functional unit groups for managers: Event Monitor, Monitoring Manager, Simple Manager, Controlling Manager, and Full Manager. Each functional unit group is a superset of the preceding group. There are five functional unit groups for agents: Event Sender, Monitored Agent, Simple Agent, Controlled Agent, and Full Agent. Again, each functional unit group is a superset of the preceding group. The operations supported for each functional unit group are summarized in Table 3.

Functional Unit Groups	Event Report	Get	Set	Create/ Delete	Action	Mult. Reply	Mult. Object Select
1. Event Monitor	U	no	no	no	no	no	no
2. Event Sender	U	no	no	no	no	no	no
3. Monitoring Mgr.	U	yes	no	no	no	no	no
4. Monitored Agent	U	yes	no	no	no	no	no
5. Simple Manager	U	yes	C	no	no	yes	no*
6. Simple Agent	U	yes	C	no	no	yes	no*
7. Controlling Mgr.	U	yes	U/C	yes	no	yes	yes
8. Controlled Agent	U	yes	U/C	yes	no	yes	yes
9. Full Manager	U/C	yes	U/C	yes	U/C	yes	yes
10. Full Agent	U/C	yes	U/C	yes	U/C	yes	yes

C = confirmed, U = non-confirmed

* Simple Managers and Agents must support "oneLevel" scoping for all and only those cases where it is required to access a whole table and may support synchronization other than "best effort"; no support for filtering is required.

Table 3. Functional Unit Groups

A conformant system must support at least one of these functional unit groups. A system may support both a manager group and an agent group. A system only needs to implement the services and service primitives required for the groups that it supports. In addition, a system may support services that are not required by any group that

it supports.

7.1.4. M-INITIALISE Parameters

The M-INITIALISE service is provided by the ACSE A-ASSOCIATE service. The parameters for the M-INITIALISE service are defined in [11] and summarized in Table 4.

Parameter Name	Req/Ind	Rsp/Conf
Functional Units	Mandatory	Mandatory
User Information	Optional	Optional
Access Control	Optional	Optional

Table 4. M-INITIALISE Parameters

Notice that the further agreement has been made that the Functional Units parameter is mandatory at all times. The M-INITIALISE parameters are conveyed as ACSE user information in the ACSE request PDU.

7.1.4.1. Functional Units

The exchange of functional units between the initiating CMISE-service-user and the responding CMISE-service-user is required. This allows the CMIS-service-users to inform each other which functional units are supported. CMIP [12] defines a 21-bit BIT STRING to communicate which functional units are supported. A functional unit is supported if the corresponding bit in this bit string is one. The correspondence between functional units and functional unit groups is given in Table 5. The left column gives the functional unit corresponding to a particular bit position. The numbers along the top of the table indicate the functional unit group (the numbers of the functional unit groups are given in Table 3). The various columns indicate the value of each bit for a particular functional unit group.

Functional Unit	1	2	3	4	5	6	7	8	9	10
conf. event report invoker(0)	0	0	0	0	0	0	0	0	0	1
conf. event report perf.(1)	0	0	0	0	0	0	0	0	1	0
event report invoker(2)	0	1	0	1	0	1	0	1	0	1
event report performer(3)	1	0	1	0	1	0	1	0	1	0
confirmed get invoker(4)	0	0	1	0	1	0	1	0	1	0
confirmed get performer(5)	0	0	0	1	0	1	0	1	0	1
confirmed set invoker(6)	0	0	0	0	1	0	1	0	1	0
confirmed set performer(7)	0	0	0	0	0	1	0	1	0	1
set invoker(8)	0	0	0	0	0	0	1	0	1	0
set performer(9)	0	0	0	0	0	0	0	1	0	1
confirmed action invoker(10)	0	0	0	0	0	0	0	0	1	0
confirmed action performer(11)	0	0	0	0	0	0	0	0	0	1
action invoker(12)	0	0	0	0	0	0	0	0	1	0
action performer(13)	0	0	0	0	0	0	0	0	0	1
confirmed create invoker(14)	0	0	0	0	0	0	1	0	1	0
confirmed create performer(15)	0	0	0	0	0	0	0	1	0	1
confirmed delete invoker(16)	0	0	0	0	0	0	1	0	1	0
confirmed delete performer(17)	0	0	0	0	0	0	0	1	0	1
multiple reply(18)	0	0	0	0	1	1	1	1	1	1
multiple object selection(19)	0	0	0	0	0	0	1	1	1	1
extended service(20)	0	0	0	0	0	0	0	0	0	0
	M	A	M	A	M	A	M	A	M	A

1 = supported, 0 = not supported, M = manager, A = agent

Table 5. Functional Unit Group Values

The "negotiation" using functional units proceeds as follows. The initiating CMISE-service-user (manager or agent) sends the functional units representing the functional unit group to which it belongs. The responding CMISE-service-user sends the functional units representing the functional unit group to which it belongs. (If an application process belongs to both a manager and an agent functional unit group, then both functional unit groups are indicated using the same functional unit bit string.) If the functional unit groups supported by the two application entities do not allow meaningful communication, then either entity may refuse the association. Meaningful communication is defined as the ability of the entity to invoke or perform at least one CMIS operation supported by the other entity (i.e., some "complementary" set of functional units exists). After an association has been established, a system must provide the proper response for functional units that it has indicated it can support and should gracefully refuse other requests in accordance

with the protocol.

7.1.4.2. User Information

The User Information parameter is optional. No entity is required to send this parameter, but all entities are expected to tolerate receipt of it.

One possible use of the User Information parameter is to convey information describing MIB extensions supported by the manager or agent. This can be viewed as a further way of refining the application context. The mechanism for doing this is not defined at this time.

7.1.4.3. Access Control

The CMIS M-INITIALISE Access Control parameter is optional. Access control is supported on a per association basis using ACSE. It is recommended (but not required) that the access control parameter be used for each A-ASSOCIATE request (via M-INITIALISE).

Access control is also possible on a per request basis with the CMIS Access Control parameter. This parameter might be used to implement security similar to the community access rights mechanism provided by SNMP [4]. It is expected that the Access Control parameter will be used to implement the standard TCP/IP authentication mechanism once this has been defined.

7.2. Supporting Services

The M-INITIALISE, M-TERMINATE, and M-ABORT services assume the use of ACSE. The following ACSE services are required: A-ASSOCIATE, A-RELEASE, A-ABORT, and A-P-ABORT. The rest of the CMIP protocol uses the RO-INVOKE, RO-RESULT, RO-ERROR, and RO-REJECT services of ROSE.

7.3. CMIP Agreements

The following sections contain specific CMIP agreements in addition to those specified in the CMIP standard [12].

7.3.1. Invoke Identifier

It is required that there be a unique invoke identifier (present in the ROSE PDU) for successive invocations on the same association. The invoke identifier is provided by the invoking CMISE-service-user. Invoke identifiers should increase monotonically during the lifetime of an association. Semantically, the invoke identifier is a Counter as defined in [2]. Unique identifiers will allow the detection of

lost and duplicate requests.

7.3.2. Object Class

The object class field of all CMIP PDUs shall be limited to the "globalForm" choice:

```
ObjectClass ::=
     CHOICE {
          globalForm    [0] IMPLICIT OBJECT IDENTIFIER
     }
```

7.3.3. Object Instance

The object instance field of all CMIP PDUs is limited to the "distinguishedName" choice:

```
ObjectInstance ::=
     CHOICE {
          distinguishedName  [2] IMPLICIT DistinguishedName
     }
```

The definition for DistinguishedName is imported from CCITT X.500 and ISO DIS 9594-2 [26]:

```
DistinguishedName ::= RDNSequence
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeValueAssertion
```

The definition for AttributeValueAssertion is contained in CMIP [12]:

```
AttributeValueAssertion ::= SEQUENCE { AttributeId, AttributeValue }
AttributeId ::=
     CHOICE {
          globalId   [0] IMPLICIT OBJECT IDENTIFIER
          localId    [1] IMPLICIT INTEGER
     }
AttributeValue ::= ANY DEFINED BY attributeId
```

Those attributes to be used as the distinguished attributes of a managed object are defined at the time of registration of the object class and are identified in the NAMES clause of the OBJECT-CLASS macro.

When there is no instance information to convey about a managed object, then the following "empty" object instance shall be used: The "distinguishedName" choice of ObjectInstance shall be an RDNSequence consisting of a SEQUENCE of one RelativeDistinguishedName. That RelativeDistinguishedName shall be an empty SET of AttributeValueAssertions.

7.3.4. Access Control

The access control parameter is optional. The receipt of this parameter must be tolerated (i.e., gracefully accepted), but a receiving entity is free to ignore this information. The Access Control field is defined in [12] as EXTERNAL. Until a more sophisticated access control mechanism is defined, simple authentication can be accomplished by using an unencrypted password in the access control field. The definition of this EXTERNAL is the same as that for the ACSE Access Control field (section 8.3.2).

7.3.5. Synchronization

Support for "best effort" synchronization is required. Atomic synchronization may also be supported, but is not required.

7.3.6. Scope

Scoping is supported if the multiple object selection functional unit is selected. If scoping is supported, all values of the scope field shall be supported.

7.3.7. Filter

Filtering is supported if the multiple object selection functional unit is selected. If filtering is supported, it is not required that all features of filtering be supported. The following are the minimal filtering requirements for any system that supports filtering. In the CMIP field CMISFilter, at least two instances of the binary operators ("and," "or") must be supported. Support for additional instances of these operators is not required. Double "not" need not be supported. In FilterItem, the arithmetic operations ("equality", "greaterOrEqual," "lessOrEqual") must be supported. The "present" choice of FilterItem must also be supported. It is not required to support string operations (namely, the "substrings" choice of the FilterItem type). Thus, the minimal requirements for filtering yield this restricted definition of FilterItem:

```
FilterItem ::=
    CHOICE {
        equality        [0] AttributeValueAssertion,
        greaterOrEqual  [2] AttributeValueAssertion,
        lessOrEqual     [3] AttributeValueAssertion,
        present         [4] AttributeID
    }
```

7.3.8. Attribute Identifier

Both choices for the CMIP AttributeId field are allowed:

```
AttributeId ::=
    CHOICE {
        globalId  [0] IMPLICIT OBJECT IDENTIFIER,
        localId   [1] IMPLICIT INTEGER
    }
```

The "globalId" form of AttributeId is required if scoping is used (i.e., the value of the scope field is other than "baseObject").

7.3.9. Event Type Identifier

Both choices for the CMIP EventTypeId field are allowed:

```
EventTypeId ::=
    CHOICE {
        globalId  [6] IMPLICIT OBJECT IDENTIFIER,
        localId   [7] IMPLICIT INTEGER
    }
```

7.3.10. Action Type Identifier

Both choices for the CMIP ActionTypeId field are allowed:

```
ActionTypeId ::=
    CHOICE {
        globalId  [2] IMPLICIT OBJECT IDENTIFIER,
        localId   [3] IMPLICIT INTEGER
    }
```

The "globalId" form of ActionTypeId is required if scoping is used (i.e., the value of the scope field is other than "baseObject").

7.3.11. Time Fields

The "eventTime" field of the m-EventReport Invoke PDU and the m-EventConfirmedReport Invoke PDU must be present.

The "currentTime" field of the following PDUs must be present: the m-EventReport Confirmed Result PDU, the m-Get Result PDU, the m-Set Result PDU, the m-Action Confirmed Result PDU, the m-Create Result PDU, the m-Delete Result PDU, the GetListError Error PDU, and the SetListError Error PDU.

All CMIP time fields shall use the ASN.1 GeneralizedTime type defined in [5] with 1 millisecond granularity.

If the system generating the PDU does not have the current time, yet does have the time since last boot, then GeneralizedTime can be used to encode this information. The time since last boot will be added to the base time "0001 Jan 1 00:00:00.00" using the Gregorian calendar algorithm. (In the Gregorian calendar, all years have 365 days except those divisible by 4 and not by 400, which have 366.) The use of the year 1 as the base year will prevent any confusion with current time.

If no meaningful time is available, then the year 0 shall be used in GeneralizedTime to indicate this fact.

7.3.12. Response PDUs

Both the "managedObjectClass" and "managedObjectInstance" fields must be present in the following CMIP response PDUs: the m-EventReport Confirmed Result PDU, the m-Get Result PDU, the m-Set Result PDU, the m-Action Confirmed Result PDU, the m-Create Result PDU, the m-Delete Result PDU, the GetListError Error PDU, and the SetListError Error PDU. The "managedObjectInstance" field must be present in the ProcessingFailure Error PDU. The "managedObjectClass" field must be present in the NoSuchArgument Error PDU.

7.3.13. Error PDUs

The "globalId" form of AttributeId is required for the NoSuchAttributeId Error PDU and the InvalidAttributeValue Error_PDU.

8. Association Control Service Element

The Association Control Service Element (ACSE), which is necessary

for establishing and releasing application associations, is defined in [7] and [8].

8.1. ACSE Services

The ACSE service description is detailed in ISO 8649 [7]. All of the defined ACSE services are mandatory:

- o A-ASSOCIATE: This confirmed service is used to initiate an application association between application entities.
- o A-RELEASE: This confirmed service is used to release an application association between application entities without loss of information.
- o A-ABORT: This unconfirmed service causes the abnormal release of an association with a possible loss of information.
- o A-P-ABORT: This provider-initiated service indicates the abnormal release of an application association by the underlying presentation service with a possible loss of information.

Mappings of the ACSE services to presentation services and ACSE APDUs are shown in Table 6, along with a section reference to ISO 8649 [7].

ACSE Service	ISO 8649 Reference	Related Presentation Service	Associated APDUs
A-ASSOCIATE	9.1	P-CONNECT	AARQ, AARE
A-RELEASE	9.2	P-RELEASE	RLRQ, RLRE
A-ABORT	9.3	P-U-ABORT	ABRT
A-P-ABORT	9.4	P-P-ABORT	(none)

Table 6. Mapping of ACSE Services

8.2. Supporting Services

ACSE will make use of the following ISO presentation layer services: P-CONNECT, P-RELEASE, P-U-ABORT, and P-P-ABORT. These presentation services will be provided by the LPP [13].

8.3. ACSE Protocol

The ACSE protocol specification is found in ISO 8650 [8]. All five ACSE APDUs specified in the standard are mandatory.

8.3.1. Application Context Name

The Application Context Name takes the form of an OBJECT IDENTIFIER. The value of this OBJECT IDENTIFIER includes both the version of CMOT being used for this association and the version number of the highest version of the Internet-standard MIB supported by the manager or agent. The application context name has the following generic form:

```
{ iso(1) org(3) dod(6) internet(1) mgmt(2) mib(n)
  cmot(9) cmotVersion(1) version-number(v) }

where n = highest MIB version supported and
      v = version of CMOT supported
```

For the version of CMOT defined in these agreements, "version-number" has the value of one (1). This version of CMOT implies the versions of the ISO protocols specified in this memo (see Figure 2).

8.3.2. User Information

The following CMIS M-INITIALISE parameters are all mapped onto the ACSE User Information parameter: Functional Units, User Information, and Access Control. (See section 7.1.4 for more information on the CMIS M-INITIALISE parameters.) ACSE User Information is defined in ISO 8650 as follows:

```
Association-information ::= SEQUENCE OF EXTERNAL
```

The ASN.1 defined type EXTERNAL, which is defined in section 35 of ISO 8824 [5], requires both an OBJECT IDENTIFIER for identification and an associated ASN.1 encoding.

The OBJECT IDENTIFIER and syntax associated with the ACSE Functional Units EXTERNAL definition are found in [12]. The OBJECT IDENTIFIER is defined as { iso(1) standard(0) ips-osi-mips(9596) cmip(2) version(1) acse(0) functional-units(0) } and the syntax is a BIT STRING.

The EXTERNAL definition for User Information is left unspecified at this time; it will be defined in a future memo.

If some form of access control is required, a simple unencrypted

password can be used. The EXTERNAL for this simple access control will use the OBJECT IDENTIFIER { cmotAcseAccessControl } (Appendix A) and the syntax OCTET STRING. A more sophisticated authentication mechanism will be defined with another EXTERNAL definition in a future memo.

8.3.3. Presentation Service Parameters

The values and defaults of parameters to the ACSE primitives that are given to the presentation service are specified in RFC 1085 [13].

For the Presentation Context Definition List parameter to the P-CONNECT service [13, p. 10], the value of the Abstract Syntax Name associated with the Presentation Context Identifier of value one (1) shall be identical to the OBJECT IDENTIFIER used for the Application Context Name (section 8.3.1).

The Quality of Service parameter shall have the value of either "tcp-based" or "udp-based."

9. Remote Operations Service Element

The Remote Operations Service Element (ROSE), which provides the ability to invoke remote operations, is specified in ISO 9072-1 [9] and 9072-2 [10]. ROSE can only be used once an association has been established between two application entities. ROSE is used to support CMISE; it is not intended to be used directly by management application processes.

9.1. ROSE Services

The ROSE service definition is detailed in ISO 9072-1 [9]. All of the defined ROSE services are mandatory:

- RO-INVOKE: This unconfirmed service is used by an invoking ROSE-user to cause the invocation of an operation to be performed by an invoked ROSE-user.

- RO-RESULT: This unconfirmed service is used by an invoked ROSE-user to reply to a previous RO-INVOKE indication in the case of a successfully performed operation.

- RO-ERROR: This unconfirmed service is used by an invoked ROSE-user to reply to a previous RO-INVOKE indication in the case of an unsuccessfully performed operation.

- RO-REJECT-U: This unconfirmed service is used by a ROSE-user to reject a request (RO-INVOKE indication) of the other

ROSE-user if it has detected a problem. It may also be used by a ROSE-user to (optionally) reject a reply (RO-RESULT indication, RO-ERROR indication) from the other ROSE-user.

- RO-REJECT-P: This provider-initiated service is used to advise a ROSE-user of a problem detected by the ROSE-provider.

Mappings of ROSE services to ISO presentation services and ROSE APDUs are shown in Table 7, along with a section reference to ISO 9072-1 [9].

ROSE Service	ISO 9072-1 Reference	Related Presentation Service	Associated APDUs
RO-INVOKE	10.1	P-DATA	ROIV
RO-RESULT	10.2	P-DATA	RORS
RO-ERROR	10.3	P-DATA	ROER
RO-REJECT-U	10.4	P-DATA	RORJ
RO-REJECT-P	10.5	P-DATA	RORJ

Table 7. Mapping of ROSE Services

9.2. Supporting Services

ROSE will only make use of the presentation layer service P-DATA. This service is provided by the LPP. The following restrictions are a consequence of the use of the LPP: First, mappings to the Reliable Transfer Service Element (RTSE) are not possible, since no RTSE is present. Second, no data token is used with the presentation services.

9.3. ROSE Protocol

The protocol specification for ROSE shall follow ISO 9072-2 [10]. All four APDUs specified in the standard are mandatory. In addition, the ability to support the correct origination and reception of the linked-id protocol element is required if the multiple reply functional unit has been selected (section 7.1.2).

9.3.1. Operation Class

Since no turn management is required by ROSE, the Operation Class parameter may be ignored.

9.3.2. Priority

ROSE will deliver each APDU in a "first in, first out" manner. Since no turn management is required by ROSE, the Priority parameter may be ignored.

10. Lightweight Presentation

The specification for the lightweight presentation protocol (LPP) is contained in RFC 1085, "ISO Presentation Services on top of TCP/IP-based internets" [13]. The services defined in that memo are the minimal set of ISO presentation services required to support ACSE and ROSE. The protocol specified to provide these services is a replacement for the ISO presentation protocol.

10.1. Lightweight Presentation Services

All of the ISO presentation services provided by the LPP are mandatory: P-CONNECT, P-RELEASE, P-U-ABORT, P-P-ABORT, and P-DATA.

10.2. Supporting Services

Depending on the quality of service indicated in the P-CONNECT request, the LPP will use either UDP (low quality) or TCP (high quality) as the underlying transport protocol. UDP provides an unreliable datagram service, while TCP provides a reliable connection-oriented transport service.

Practically speaking, there are two ways to discover whether a remote system supports the LPP over UDP or TCP. The first is to use some undefined form of directory service. This might be nothing more than a local table. The second way is simply to attempt to establish an association with the remote application entity using the desired quality of service. If the transport for that service is unavailable on the remote system, then the local presentation-service-provided will issue a negative P-CONNECT.CONFIRMATION primitive. This will be interpreted by ACSE as a failure to establish an association with the desired quality of service.

The following well-known UDP and TCP port numbers are defined:

```
cmot manager    163/tcp
cmot manager    163/udp
cmot agent      164/tcp
cmot agent      164/udp
```

When UDP is used, an implementation need not accept a lightweight presentation PDU whose length exceeds 484. The purpose of this

restriction is to ensure that CMIP requests and responses can be transmitted in a single unfragmented IP datagram.

10.3. Lightweight Presentation Protocol

No further agreements are needed for the lightweight presentation protocol defined in RFC 1085.

11. Acknowledgements

This RFC is the work of many people. The following members of the IETF Netman working group and other interested individuals made important contributions:

Amatzia Ben-Artzi, 3Com
Asheem Chandna, AT&T Bell Laboratories
Ken Chapman, Digital Equipment Corporation
Anthony Chung, Sytek
George Cohn, Ungermann-Bass
Gabriele Cressman, Sun Microsystems
Pranati Kapadia, Hewlett-Packard
Lee LaBarre, The MITRE Corporation (chair)
Dave Mackie, 3Com
Keith McCloghrie, The Wollongong Group
Jim Robertson, 3Com
Milt Roselinsky, CMC
Marshall Rose, The Wollongong Group
John Scott, Data General
Lou Steinberg, IBM

12. References

[1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, April 1988.

[2] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1065, August 1988.

[3] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1066, August 1988.

[4] Case, J., M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)", RFC 1098, (Obsoletes RFC 1067), April 1989.

[5] ISO 8824: "Information processing systems - Open Systems

Interconnection, Specification of Abstract Syntax Notation One (ASN.1)", Geneva, March 1988.

[6] ISO 8825: "Information processing systems - Open Systems Interconnection, Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)", Geneva, March 1988.

[7] ISO 8649: "Information processing systems - Open Systems Interconnection, Service Definition for Association Control Service Element".

[8] ISO 8650: "Information processing systems - Open Systems Interconnection, Protocol Specification for Association Control Service Element".

[9] CCITT Recommendation X.219, Working Document for ISO 9072-1: "Information processing systems - Text Communication, Remote Operations: Model, Notation and Service Definition", Gloucester, November 1987.

[10] CCITT Recommendation X.229, Working Document for ISO 9072-2: "Information processing systems - Text Communication, Remote Operations: Protocol Specification", Gloucester, November 1987.

[11] ISO DIS 9595-2: "Information processing systems - Open Systems Interconnection, Management Information Service Definition - Part 2: Common Management Information Service", 22 December 1988.

[12] ISO DIS 9596-2: "Information Processing Systems - Open Systems Interconnection, Management Information Protocol Specification - Part 2: Common Management Information Protocol", 22 December 1988.

[13] Rose, M., "ISO Presentation Services on top of TCP/IP-based internets", RFC 1085, December 1988.

[14] OSI Network Management Forum, "Forum Interoperable Interface Protocols", September 1988.

[15] ISO DIS 7498-4: "Information processing systems - Open Systems Interconnection, Basic Reference Model - Part 4: OSI Management Framework".

[16] ISO/IEC JTC1/SC21/WG4 N571: "Information processing systems - Open Systems Interconnection, Systems Management: Overview", London, July 1988.

[17] Klerer, S. Mark, "The OSI Management Architecture: An Overview", IEEE Network Magazine, March 1988.

[18] Ben-Artzi, A., "Network Management for TCP/IP Networks: An Overview", Internet Engineering Task Force working note, April 1988.

[19] ISO/IEC JTC1/SC21/WG4 N3324: "Information processing systems - Open Systems Interconnection, Management Information Services - Structure of Management Information - Part I: Management Information Model", Sydney, December 1988.

[20] Postel, J., "User Datagram Protocol", RFC 768, August 1980.

[21] Postel, J., "Transmission Control Protocol", RFC 793, September 1981.

[22] ISO DP 9534: "Information processing systems - Open Systems Interconnection, Application Layer Structure", 10 March 1987.

[23] Rose, M., "ISO Transport Services on top of the TCP", RFC 1006, May 1987.

[24] ISO 8822: "Information processing systems - Open Systems Interconnection, Connection Oriented Presentation Service Definition", June 1987.

[25] Postel, J., "Internet Protocol", RFC 791, September 1981.

[26] CCITT Draft Recommendation X.500, ISO DIS 9594/1-8: "The Directory", Geneva, March 1988.

Appendix A - The CMOT Group

```
   CMOT DEFINITIONS ::= BEGIN

   IMPORTS OBJECT-TYPE FROM RFC1065-SMI;

   IMPORTS mib FROM RFC1066-MIB;

     cmot  OBJECT IDENTIFIER ::= { mib 9 }

     -- The following assignments are made for the purpose of
     -- identification within CMOT and do not refer to MIB objects.

     cmotVersion                OBJECT IDENTIFIER ::= { cmot 1 }

     cmotAcseInfo               OBJECT IDENTIFIER ::= { cmot 2 }
     cmotAcseAccessControl      OBJECT IDENTIFIER ::= { cmotAcseInfo 1 }

     -- The following definition is made for use in referencing a
     -- managed system (for the purpose of proxy management) in the
     -- CMIP Object Instance field. It does not represent a MIB
     -- object.

     cmotSystemID OBJECT-TYPE
             SYNTAX  CmotSystemID
             ACCESS  not-accessible
             STATUS  optional
             ::= { cmot 3 }

     CmotSystemID ::= CHOICE {
             arbitrary       [0] IMPLICIT OCTET STRING,
             proxyIndex      [1] IMPLICIT INTEGER,
             inetAddr        [2] IMPLICIT IpAddress,
             domainName      [3] IMPLICIT OCTET STRING,
             mac802Addr      [4] IMPLICIT OCTET STRING,
             x121Addr        [5] IMPLICIT OCTET STRING,
             nsap            [6] IMPLICIT OCTET STRING,
             netbiosName     [7] IMPLICIT OCTET STRING,
             snaName         [8] IMPLICIT OCTET STRING,
             adminId         [9] IMPLICIT OBJECT IDENTIFIER
     }

      -- All addresses should be conveyed in network-byte order.

   END
```

Appendix B - Management Information Summary

```
RFC1066-MIB-INTERPRETATION

        { iso org(3) dod(6) internet(1) mgmt(2) 1 }

            DEFINITIONS ::= BEGIN

            IMPORTS mgmt, OBJECT-TYPE FROM RFC1065-SMI;

              mib           OBJECT IDENTIFIER ::= { mgmt 1 }

              system        OBJECT IDENTIFIER ::= { mib 1 }
              interfaces    OBJECT IDENTIFIER ::= { mib 2 }
              at            OBJECT IDENTIFIER ::= { mib 3 }
              ip            OBJECT IDENTIFIER ::= { mib 4 }
              icmp          OBJECT IDENTIFIER ::= { mib 5 }
              tcp           OBJECT IDENTIFIER ::= { mib 6 }
              udp           OBJECT IDENTIFIER ::= { mib 7 }
              egp           OBJECT IDENTIFIER ::= { mib 8 }


        -- definition of object class

        OBJECT-CLASS MACRO  ::=
        BEGIN
          TYPE NOTATION  ::= SubClassOf Superiors Names Attributes
          VALUE NOTATION ::= value(VALUE OBJECT IDENTIFIER)

          SubClassOf     ::= "SUBCLASS OF" value(OBJECT-CLASS)
                           | empty
          Superiors      ::= "SUPERIORS" "{" SuperiorList "}"
                           | empty
          Names          ::= "NAMES" "{" AttributeList "}"
                           | empty
          Attributes     ::= "CONTAINS" "{" AttributeList "}"
                           | empty

          SuperiorList   ::= Superior | Superior "," SuperiorList
          Superior       ::= value(OBJECT-CLASS)

          AttributeList  ::= Attribute | Attribute "," AttributeList
          Attribute      ::= value(OBJECT-TYPE)

        END

        -- the System group
```

```
system OBJECT-CLASS
        NAMES  { cmotSystemID }    -- Appendix A
        CONTAINS  {
                sysDescr,
                sysObjectID,
                sysUpTime
        }
        ::= { mib 1 }

-- the Interfaces group

interfaces OBJECT-CLASS
        SUPERIORS  { system }
        CONTAINS  { ifNumber }
        ::= { mib 2 }

ifTable OBJECT-CLASS
        SUPERIORS  { interfaces }
        ::= { interfaces 2 }

ifEntry OBJECT-CLASS
        SUPERIORS  { ifTable }
        NAMES { ifIndex }
        CONTAINS  {
                ifIndex,
                ifDescr,
                ifType,
                ifMtu,
                ifSpeed,
                ifPhysAddress,
                ifAdminStatus,
                ifOperStatus,
                ifLastChange,
                ifInOctets,
                ifInUcastPkts,
                ifInNUcastPkts,
                ifInDiscards,
                ifInErrors,
                ifInUnknownProtos,
                ifOutOctets,
                ifOutUcastPkts,
                ifOutNUcastPkts,
                ifOutDiscards,
                ifOutErrors,
                ifOutQLen
        }
        ::= { ifTable 1 }
```

```
-- the Address Translation group

at OBJECT-CLASS
        SUPERIORS  { system }
        ::= { mib 3 }

atTable OBJECT-CLASS
        SUPERIORS  { at }
        ::= { at 1 }

atEntry OBJECT-CLASS
        SUPERIORS  { atTable }
        NAMES  {
                atIfIndex,
                atNetAddress
        }
        CONTAINS  {
                atIfIndex,
                atPhysAddress,
                atNetAddress
        }
        ::= { atTable 1 }

-- the IP group

ip OBJECT-CLASS
SUPERIORS  { system }
CONTAINS  {
                ipForwarding,
                ipDefaultTTL,
                ipInReceives,
                ipInHdrErrors,
                ipInAddrErrors,
                ipForwDatagrams,
                ipInUnknownProtos,
                ipInDiscards,
                ipInDelivers,
                ipOutRequests,
                ipOutDiscards,
                ipOutNoRoutes,
                ipReasmTimeout,
                ipReasmReqds,
                ipReasmOKs,
                ipReasmFails,
                ipFragOKs,
                ipFragFails,
                ipFragCreates
}
```

```
        ::= { mib 4 }

-- the IP Interface table

ipAddrTable OBJECT-CLASS
        SUPERIORS  { ip }
        ::= { ip 20 }

ipAddrEntry OBJECT-CLASS
        SUPERIORS  { ipAddrTable }
        NAMES  { ipAdEntAddr }
        CONTAINS  {
                ipAdEntAddr,
                ipAdEntIfIndex,
                ipAdEntNetMask,
                ipAdEntBcastAddr
        }
        ::= { ipAddrTable 1 }

-- the IP Routing table

ipRoutingTable OBJECT-CLASS
        SUPERIORS  { ip }
        ::= { ip 21 }

ipRouteEntry OBJECT-CLASS
        SUPERIORS  { ipRoutingTable }
        NAMES  { ipRouteDest }
        CONTAINS  {
                ipRouteDest,
                ipRouteIfIndex,
                ipRouteMetric1,
                ipRouteMetric2,
                ipRouteMetric3,
                ipRouteMetric4,
                ipRouteNextHop,
                ipRouteType,
                ipRouteProto,
                ipRouteAge
        }
        ::= { ipRoutingTable 1 }

-- the ICMP group

icmp OBJECT-CLASS
        SUPERIORS  { system }
        CONTAINS  {
                icmpInMsgs,
```

```
                        icmpInErrors,
                        icmpInDestUnreachs,
                        icmpInTimeExcds,
                        icmpInParmProbs,
                        icmpInSrcQuenchs,
                        icmpInRedirects,
                        icmpInEchos,
                        icmpInEchoReps,
                        icmpInTimestamps,
                        icmpInTimestampReps,
                        icmpInAddrMasks,
                        icmpInAddrMaskReps,
                        icmpOutMsgs,
                        icmpOutErrors,
                        icmpOutDestUnreachs,
                        icmpOutTimeExcds,
                        icmpOutParmProbs,
                        icmpOutSrcQuenchs,
                        icmpOutRedirects,
                        icmpOutEchos,
                        icmpOutEchoReps,
                        icmpOutTimestamps,
                        icmpOutTimestampReps,
                        icmpOutAddrMasks,
                        icmpOutAddrMaskReps
                }
                ::= { mib 5 }

        -- the TCP group

        tcp OBJECT-CLASS
                SUPERIORS  { system }
                CONTAINS  {
                        tcpRtoAlgorithm,
                        tcpRtoMin,
                        tcpRtoMax,
                        tcpMaxConn,
                        tcpActiveOpens,
                        tcpPassiveOpens,
                        tcpAttemptFails,
                        tcpEstabResets,
                        tcpCurrEstab,
                        tcpInSegs,
                        tcpOutSegs,
                        tcpRetransSegs
                }
                ::= { mib 6 }
```

```
        -- the TCP connections table

        tcpConnTable OBJECT-CLASS
                SUPERIORS  { tcp }
                ::= { tcp 13 }

        tcpConnEntry OBJECT-CLASS
                SUPERIORS  { tcpConnTable }
                NAMES  {
                        tcpConnLocalAddress,
                        tcpConnLocalPort,
                        tcpConnRemAddress,
                        tcpConnRemPort
                }
                CONTAINS  {
                        tcpConnState,
                        tcpConnLocalAddress,
                        tcpConnLocalPort,
                        tcpConnRemAddress,
                        tcpConnRemPort
                }
                ::= { tcpConnTable 1 }

        -- the UDP group

        udp OBJECT-CLASS
                SUPERIORS  { system }
                CONTAINS  {
                        udpInDatagrams,
                        udpNoPorts,
                        udpInErrors,
                        udpOutDatagrams
                }
                ::= { mib 7 }


        -- the EGP group

        egp OBJECT-CLASS
                SUPERIORS  { system }
                CONTAINS  {
                        egpInMsgs,
                        egpInErrors,
                        egpOutMsgs,
                        egpOutErrors
                }
                ::= { mib 8 }
```

```
        -- the EGP Neighbor table

        egpNeighTable OBJECT-CLASS
                SUPERIORS  { egp }
                ::= { egp 5 }

        egpNeighEntry OBJECT-CLASS
                SUPERIORS  { egpNeighTable }
                NAMES  { egpNeighAddr }
                CONTAINS  {
                        egpNeighState,
                        egpNeighAddr
                }
                ::= { egpNeighTable 1 }


        END
```

Appendix C - Sample Protocol Exchanges

The following are sample protocol exchanges between a manager and an agent. The manager establishes an association with the agent, requests the number of IP address and header errors, requests the type of route corresponding to the destination address 10.0.0.51, requests the TCP connection with the well-known port for FTP, and then releases the association. All of these samples show the lightweight presentation protocol being used over TCP.

```
--
-- the manager sends an ACSE association request carried in a
-- presentation connect request PDU
--

{
   connectRequest {                                    -- LPP
      version version-1,
      reference {
         callingSSUserReference "sri-nic.arpa",
         commonReference "880821222531Z"
      },
      asn 1.3.6.1.2.1.9.1.1,
      user-data {                                      -- ACSE
         protocol-version version1,
         application-context-name 1.3.6.1.2.1.9.1.1,
         user-information {
            functionalUnits {
               direct-reference 1.0.9596.2.1.0.0,
               encoding {
                  single-ASN1-type '01011010101010101010110B'
                                                       -- Full Manager
               }
            }
         }
      }
   }
}


--
-- the agent sends an ACSE association response carried in a
-- presentation connect response PDU
--

{
   connectResponse {                                   -- LPP
      user-data {
```

```
            user-information {                          -- ACSE
               functionalUnits {
                  direct-reference 1.0.9596.2.1.0.0,
                  encoding {
                     single-ASN1-type '10100101010101010110B'
                                                          -- Full Agent
                  }
               }
            }
         }
      }

      --
      -- the manager sends a get request to read the values of
      -- ipInHdrErrors and ipInAddrErrors
      --

      {
         userData {                                     -- LPP
            ro-Invoke {                                 -- ROSE
               invokeID 10,
               operation-value m-Get(3),
               argument {                               -- CMIP
                  baseManagedObjectClass {
                     globalForm ip { 1.3.6.1.2.1.4 }
                  },
                  baseManagedObjectInstance {
                     distinguishedName {
                        relativeDistinguishedName {}
                     }
                  },
                  attributeIdList {
                     attributeId {
                        localID 4                       -- ipInHdrErrors
                     },
                     attributeId {
                        localID 5                       -- ipInAddrErrors
                     }
                  }
               }
            }
         }
      }
```

```
   --
   -- the agent replies with a get response indicating that
   -- ipInHdrErrors = 0 and ipInAddrErrors = 2
   --

   {
      userData {                                          -- LPP
         ro-Result {                                      -- ROSE
            invokeID 10,
            {
               operation-value m-Get(3),
               argument {                                 -- CMIP
                  baseManagedObjectClass {
                     globalForm ip { 1.3.6.1.2.1.4 }
                  },
                  baseManagedObjectInstance {
                     distinguishedName {
                        relativeDistinguishedName {}
                     }
                  },
                  currentTime "19880821222541.300000Z",
                  attributeList {
                     attribute {
                        attributeId {
                           localID 4                     -- ipInHdrErrors
                        },
                        attributeValue 0
                     },
                     attribute {
                        attributeId {
                           localID 5                     -- ipInAddrErrors
                        },
                        attributeValue 2
                     }
                  }
               }
            }
         }
      }

   --
   -- the manager sends a get request to discover the ipRouteType for
   -- the IP routing entry with ipRouteDest = 10.0.0.51
   --
```

```
{
   userData {                                          -- LPP
      ro-Invoke {                                      -- ROSE
         invokeID 11,
         operation-value m-Get (3),
         argument {                                    -- CMIP
            baseManagedObjectClass {
               globalForm ipRouteEntry { 1.3.6.1.2.1.4.21.1 }
            },
            baseManagedObjectInstance {
               distinguishedName {
                  relativeDistinguishedName {
                     attributeValueAssertion {
                        attributeType ipRouteDest
                                       { 1.3.6.1.2.1.4.21.1.1 },
                        attributeValue 10.0.0.51
                     }
                  }
               }
            },
            attributeIdList {
               attributeId {
                  localID 8                            -- ipRouteType
               }
            }
         }
      }
   }
}

--
-- the agent replies with a get response indicating the appropriate
-- route type
--

{
   userData {                                          -- LPP
      ro-Result {                                      -- ROSE
         invokeID 11,
         {
            operation-value m-Get(3),
            argument {                                 -- CMIP
               baseManagedObjectClass {
                  globalForm ipRouteEntry { 1.3.6.1.2.1.4.21.1 }
               },
               baseManagedObjectInstance {
                  distinguishedName {
```

```
                        relativeDistinguishedName {
                           attributeValueAssertion {
                              attributeType ipRouteDest
                                            { 1.3.6.1.2.1.4.21.1.1 },
                              attributeValue 10.0.0.51
                           }
                        }
                     }
                  },
                  currentTime "19880821222613.780000Z",
                  attributeList {
                     attribute {
                        attributeId {
                           localID 8                    -- ipRouteType
                        },
                        attributeValue "direct"
                     }
                  }
               }
            }
         }
      }

--
-- the manager sends a get request to read the TCP connection with
-- the well-known port for FTP.
--

{
   userData {                                           -- LPP
      ro-Invoke {                                       -- ROSE
         invokeID 12,
         operation-value m-Get(3),
         argument {                                     -- CMIP
            baseManagedObjectClass {
               globalForm tcpConnTable { 1.3.6.1.2.1.6.13 }
            },

            baseManagedObjectInstance {
               distinguishedName {
                  relativeDistinguishedName { }
               }
            },
            scope oneLevel(1),
            filter {
               item {
```

```
                    equality {
                        attributeType tcpConnLocalPort
                              { 1.3.6.1.2.1.6.13.1.3 }
                        attributeValue 21           -- ftp
                    }
                }
            }
            attributeIdList { } -- an empty list means all attributes
        }
    }
}


--
-- the agent replies with a get response providing the desired TCP
-- connection information. If more than one TCP connection had
-- satisfied the filter condition, a series of one or more linked
-- reply PDUs would have been returned before the final get response.
--

{
   userData {                                          -- LPP
      ro-Result {                                      -- ROSE
         invokeID 12,
         {
            operation-value m-Get(3),
            argument {                                 -- CMIP
               baseManagedObjectClass {
                  globalForm tcpConnEntry { 1.3.6.1.2.1.6.13.1 }
               },
               baseManagedObjectInstance {
                  distinguishedName {
                     relativeDistinguishedName {
                        attributeValueAssertion {
                           attributeType  { tcpConnLocalAddress },
                           attributeValue 128.10.0.34
                        },
                        attributeValueAssertion {
                           attributeType  { tcpConnLocalPort },
                           attributeValue 21
                        },
                        attributeValueAssertion {
                           attributeType  { tcpConnRemAddress },
                           attributeValue 0.0.0.0
                        },
                        attributeValueAssertion {
                           attributeType  { tcpConnRemPort },
```

```
                    attributeValue 0
                },
            }
        }
    },
    currentTime "19880821222541.300000Z",
    attributeList {
        attribute {
            attributeId {
                localId 1                   -- tcpConnState
            },
            attributeValue LISTEN
        },
        attribute {
            attributeId {
                localId 2                   -- tcpConnLocalAddress
            },
            attributeValue 128.10.0.34
        },
        attribute {
            attributeId {
                localId 3                   -- tcpConnLocalPort
            },
            attributeValue 21
        },
        attribute {
            attributeId {
                localId 4                   -- tcpConnRemAddress
            },
            attributeValue 0.0.0.0
        },
        attribute {
            attributeId {
                localId 5                   -- tcpConnRemPort
            },
            attributeValue 0
        }
    }
}
```

```
   --
   -- the manager sends a presentation release request
   --

   {
      releaseRequest {                                    -- LPP
         user-data {                                      -- ACSE
            reason normal
         }
      }
   }


   --
   -- the agent sends a presentation release response
   --

   {
      releaseResponse {                                   -- LPP
         user-data {                                      -- ACSE
            reason normal
         }
      }
   }
```

Authors' Addresses

Unnikrishnan S. Warrier
Unisys Corporation
2400 Colorado MS #42-13
Santa Monica, CA 90406

Phone: (213) 453-5196

Email: unni@cs.ucla.edu

Larry Besaw
Hewlett-Packard
3404 East Harmony Road
Fort Collins, CO 80525

Phone: (303) 229-6022

Email: lmb%hpcndaw@hplabs.hp.com